UNIVERSIDAD CARLOS III DE MADRID

TRABAJO FIN DE GRADO

RECONOCIMIENTO BIOMÉTRICO VASCULAR Y SU EVALUACIÓN DE RENDIMIENTO EN PLATAFORMA BIOAPI C#

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: Stanislav Hryhor

Tutor: Raúl Sánchez Reíllo

Leganés, 19 de Septiembre de 2014

Agradecimientos

Quiero agradecer toda la ayuda que he recibido durante el transcurso de este TFG, a toda la gente del Grupo Universitario de Tecnologías de Identificación y especialmente a Rubén quien me ha animado y ayudado cuando ya me quedaban pocas ganas de seguir.

Igualmente tengo que agradecer el apoyo constante de mis padres, Taras y Ella, y el de mis amigos, especialmente el de Carlos, por estar ahí siempre que lo he necesitado.

Resumen

La identificación biométrica es aquella que se basa en el uso de las características físicas o de comportamiento de las personas. En las décadas finales del siglo XX y en los inicios del XXI el uso de sistemas automáticos capaces de confirmar identidades de personas por medio de las técnicas biométricas ha cobrado cada vez más importancia. Sus bajísimas tasas de error y posibilidad de fraude han hecho que importantes entidades como los órganos de seguridad del estado y las entidades bancarias, se hallan fijado en estos sistemas y los implementen en sus actividades con cada vez mayor frecuencia.

Tal es el interés que despierta la biometría que ha alentado el desarrollo de muy diversas técnicas de identificación en grupos de investigación por todo el mundo. Estas técnicas van desde la clásica identificación por huella dactilar hasta la identificación por árbol vascular de las manos pasando por identificaciones tan novedosas como las dinámicas de tecleo en un teclado de ordenador.

Estando esta industria en pleno desarrollo, igualmente lo está la estandarización dentro de ella. Actualmente existen y están en continuo proceso de mejora distintas normas de estandarización, como el PIV (creado y promovido por NIST y el FBI) o BioAPI (creado y promovido por ANSI y BioAPI Consortium).

En este Trabajo de Fin de Grado (TFG) se ha traducido un algoritmo de reconocimiento biométrico vascular desde el lenguaje M (MATLAB) al lenguaje C#, con el uso de bibliotecas de código libre Emgu CV. Además este algoritmo ha sido adaptado al estándar BioAPI a fin de demostrar la posibilidad de estandarización de algoritmos que usan código libre en este lenguaje. Por último, se ha creado una aplicación, en este mismo lenguaje (C#), para poder comprobar el funcionamiento del algoritmo traducido y adaptado al estándar.

En la memoria de este TFG se familiariza al lector con las herramientas utilizadas durante el trabajo y se describe el proceso de traducción y adaptación al estándar del algoritmo.

Abstract

The biometric identification uses human physical o behavioural characteristics to confirm individual's identity. Due to its remarkably low error rate and possibility of fraud, the use of automatic systems able to perform these identifications has increased dramatically in the last decades.

The existent interest in this technology has encouraged the development of a variety of identification techniques. Now, important institutions such as the bank industry or the organizations of national security are very interested and contribute in the development of this technology. At the same time others, like NIST, ANSI or BioAPI consortium are developing standards for this growing industry.

In this Final Degree Project, an algorithm for vascular biometric identification has been translated from the M language (MATLAB) to C# Language. Using open source libraries, Emgu CV, this algorithm has been translated and adapted to the BioAPI standard. Finally, there was created an application, in order to check the performance of the translated algorithm and its adaptation to the standard.

In this document, the reader will learn about the tools, which were used during the project, the process of translation of the algorithm and its adaptation to the standard.

Índice

Índice de Tablas

Listado de Acrónimos

AHE	Adaptive Histogram Equalization
ANSI	American National Standards Institute
API	Aplication Programming Interface
BIR	Biometric Identification Record
BSP	Biometric Service Provider
CLAHE	Contrast-Limited Adaptive Histogram Equalization
CLR	Common Language Runtime
EER	Equal Error rate
FNMR	False Non-Match Rate
FMR	False Match Rate
GUTI	Grupo Universitario de Tecnologías de Identificación
IEC	International Electrotechnical Comission
ISO	International Organization for Standarization
ROI	Region of Interest (Región de Interés)
SPI	Service Provider Interface
TFG	Trabajo Fin de Grado
UC3M	Universidad Carlos III de Madrid
UUID	Universally Unique Identifier

1 Introducción

En la presente memoria se va a describir el TFG (Trabajo de Fin de Grado) realizado cuyo objetivo era la traducción de un algoritmo de identificación biométrica vascular desde el lenguaje MATLAB al lenguaje C#. Esto se ha realizado utilizando bibliotecas de tratamiento de imágenes de código libre (en concreto la Emgu CV), y al mismo tiempo se ha realizado la adaptación de este algoritmo al estándar BioAPI (ISO/IEC 19784-1) para la plataforma C# (el cual se está definiendo como norma ISO/IEC 30106-3). Para comprobar la correcta implementación y adaptación del código, se ha creado una aplicación capaz de reclutar, autenticar e identificar usuarios utilizando dicho código. La utilidad de este trabajo es demostrar que es posible la implantación de sistemas de identificación vascular que utilicen código libre y a la vez puedan cumplir el estándar.

1.1 Motivación

Desde que fue creada y hasta hoy día, la modalidad de identificación biométrica por huella dactilar es la más estudiada y desarrollada que existe. Sin embargo no es la más fiable (por ejemplo las pruebas de identidad por ADN son más fiables), ni tampoco la más segura frente al robo (por ejemplo, los patrones biométricos internos como el patrón vascular o el ADN son más complicados de robar). Entre todos los parámetros utilizados para analizar una modalidad biométrica, universalidad, unicidad, etc., a menudo el compromiso entre la fiabilidad y la facilidad de captura ha espoleado la búsqueda y desarrollo de nuevas modalidades de identificación con el fin de encontrar una con gran fiabilidad y simultáneamente una amplia aceptación[1].

Las modalidades de retina y ADN mencionadas son considerablemente más fiables que la huella dactilar, pero entran frontalmente en conflicto con lo que el usuario medio está dispuesto a hacer (laser enfocando directamente el ojo o muestras de tejidos o secreciones corporales) para conseguir una inequívoca identificación. En este aspecto, la modalidad de identificación vascular cumple mejor que ninguna otra con el compromiso mencionado, ofreciendo tasas de error bajas (aunque no tanto como las de la huella dactilar) y simultáneamente consigue mantener una mejor aceptación por el gran público, así como un alto grado de seguridad al tratarse de información interna del usuario.

Es por ello que la motivación para este trabajo ha sido el desarrollo de una modalidad innovadora y que por los aspectos descritos tiene gran futuro comercial como ya demuestra su implementación en el sistema bancario japonés [2] y británico[3].

1.2 Beneficios Sociales y Legislación referentes a la Biometría

En pocas décadas los sistemas de seguridad que produce la industria biométrica se han vuelto indispensables. Las situaciones en las que es necesaria una inequívoca identificación (por ejemplo identificación de personas en las fronteras) y los escenarios en los que la seguridad es componente fundamental (por ejemplo las sucursales bancarias) han espoleado a la industria en la creación de sistemas cada vez más elaborados y seguros.

La introducción de sistemas biométricos de seguridad modernos supone un conjunto de beneficios suficiente como para alentar su creciente implantación en organismos tanto públicos como privados. Los beneficios, fruto de estos sistemas, son importantes tanto para estas entidades como para los usuarios finales de los sistemas. Para estas entidades las ventajas de los sistemas biométricos suponen una reducción del fraude y aumento de la seguridad (impidiendo robos de credenciales o el cese voluntario de estas entre empleados). Los beneficios incluyen también la reducción de costes de mantenimiento y mejora de la imagen corporativa que supone la introducción de tecnología puntera en la empresa. A la vez, los usuarios finales también encuentran beneficiosos estos sistemas por varios motivos, que van desde mayor comodidad (con estos sistemas no es necesario recordar largas contraseñas de caracteres aleatorios y que además deben cambiar cada poco tiempo por motivos de seguridad) hasta una mayor sensación de seguridad al reducirse la posibilidad de suplantación de identidad.

Como cualquier aspecto social, la biometría necesita de normas que regulen los requisitos de los sistemas biométricos. Los datos biométricos se consideran de carácter personal y por lo tanto su tratamiento debe estar regulado a fin de preservar la privacidad de los usuarios. Para la regulación del tratamiento de estos datos, desde la Unión Europea, se han dictado directivas que obligan a los estados miembros a legislar de un modo determinado sobre estos aspectos.

> **Directiva relativa al tratamiento y circulación de datos personales a fin de proteger a las personas físicas – Directiva 95/46/CE, del Parlamento Europeo y del Consejo, de 24 de octubre[4].**

Esta directiva se dictó con el objetivo fundamental de que todos los estados miembro garanticen adecuadamente el derecho a la intimidad en lo referido al tratamiento de datos personales. El artículo 2 a) de esta directiva define de esta manera los datos personales:

> "*Toda información sobre una persona física identificada o identificable (el 'interesado'); se considerará identificable toda persona cuya identidad pueda determinarse, directa o indirectamente, en particular mediante un número de identificación o uno o varios elementos específicos, característicos de su identidad física, fisiológica, psíquica, económica, cultural o social.*"

De esta definición se deriva que los datos biométricos (como se indica arriba) son considerados datos personales y por ello se aplicara a ellos la legislación pertinente.

En España la directiva europea se adaptó a través de las siguientes leyes y decretos:

> **Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal[5] y Real Decreto 1720/2007, de 21 de diciembre, por el que se aprueba el Reglamento de Desarrollo de la LOPD[6]**

En estas normativas se establecen los principios de obligado cumplimiento que deberá seguir toda entidad que trate con datos de carácter personal durante el desarrollo de su actividad. Son de especial interés los siguientes principios mencionados en las directivas:

- Deber de información a los afectados
- Deber de obtención del consentimiento del interesado
- Principio de calidad de los datos
- Respuesta al ejercicio de los derechos de acceso, rectificación, cancelación y oposición por parte de los interesados (derechos ARCO)

En el ámbito de la biometría es especialmente destacable el artículo 6.1 de la LOPD en el que se establece la necesidad del consentimiento del interesado para el tratamiento de sus datos [7].

1.3 Objetivos

Al hablar de los objetivos es conveniente separarlos en dos partes: los que conforman el trabajo, los requisitos y especificaciones que se han de cumplir, y los perseguidos por el alumno por medio de la realización del trabajo.

En el terreno de los requisitos, el primer objetivo ha sido trasladar el código del algoritmo de identificación vascular a C# desde MATLAB. Se le pide al alumno que haga esta "traducción" ya que el lenguaje C# está diseñado para crear aplicaciones visuales capaces de ofrecer al usuario una interacción dinámica e intuitiva al contrario que en MATLAB, lenguaje

principalmente interesado en facilitar la especificación de operaciones matemáticas, por encima de la comodidad del usuario o de la eficiencia computacional (especialmente en tiempo de cálculo).

El segundo objetivo fue adaptar el código portado al estándar BioAPI en versión C# (ISO/IEC 30106-3) con el fin de una fácil sustitución futura de fragmentos del algoritmo, así como su correcto funcionamiento con distintos sensores proveedores de muestras vasculares, que son algunas de las características que ofrece dicho estándar.

Por último, el tercer objetivo fue crear una aplicación capaz de utilizar el algoritmo traducido por medio de librerías de código libre.

En cuanto al aprendizaje del alumno, los objetivos perseguidos durante la realización del trabajo eran dos. El primero, familiarizarse con los lenguajes de programación utilizados en el proyecto, completamente nuevos para el alumno. La obligación de aprender a trabajar con estas herramientas desde cero es la mejor manera de asegurar que el alumno domina las bases de las mismas, ya que sin ellas no se podría avanzar en el trabajo.

Como segundo objetivo, ha sido clave la introducción del alumno en el tipo de trabajo que uno se encuentra en el mundo laboral: los proyectos. A lo largo de la realización de los estudios de grado, excepto pequeñas imitaciones en algunas asignaturas, no se han realizado proyectos de envergadura, y es por ello que este TFG ha sido la primera toma de contacto del alumno con lo que se encontrara en el mundo laboral.

1.4 Estructura

En el presente documento se realiza una introducción a la tecnología con el análisis del estado del arte y las tecnologías asociadas. El apartado de Tecnologías Asociadas incluye una descripción del lenguaje C#, las librerías EMGU CV utilizadas y una breve exposición sobre el estándar BioAPI. Los siguientes apartados incluyen el enfoque de los objetivos del proyecto y como se ha llegado a la solución final, así como la descripción detallada del proceso de trabajo con los distintos problemas, y soluciones, surgidos durante la realización de este. A continuación se exponen los resultados obtenidos en las pruebas a las que ha sido sometida la aplicación final y una breve conclusión sobre estos que incluye el peso de este trabajo en las líneas de investigación futuras.

Y por último pero no menos importante, se exponen en el Anexo A las fases de todo el trabajo y el presupuesto resultante de la realización de este TFG.

2 Estado del Arte

En este capítulo se introducirá al lector en la biometría. Se realizará un breve resumen de su historia, se analizarán las partes de las que se compone un sistema biométrico convencional y se comentarán los diversos métodos biométricos.

2.1 Historia de la Biometría

La biometría es la ciencia que estudia los métodos de registro e identificación de seres humanos mediante el reconocimiento de características únicas biológicas, físicas o conductuales. Se sabe que los comerciantes chinos utilizaban las estampas de las huellas de las manos en papel con tinta como método de identificación ya en el siglo XIV[8].

La identificación biométrica no se empezó a utilizar en occidente hasta el siglo XIX, que fue cuando comenzó su uso con fines policiales. Al principio la identificación se llevaba a cabo de manera muy rudimentaria, confiando exclusivamente en la memoria, hasta que en 1883 Alphonse Bertillon, destacado médico y estadístico y también miembro de la policía parisina, desarrolló su sistema antropométrico de clasificación. Este método fue el primer sistema preciso y ampliamente utilizado para la identificación de criminales. Incluía, entre otras, las medidas de largo y ancho de la cabeza, largo del pie izquierdo o el largo del antebrazo del individuo, al igual que el registro de marcas personales tales como tatuajes o cicatrices.

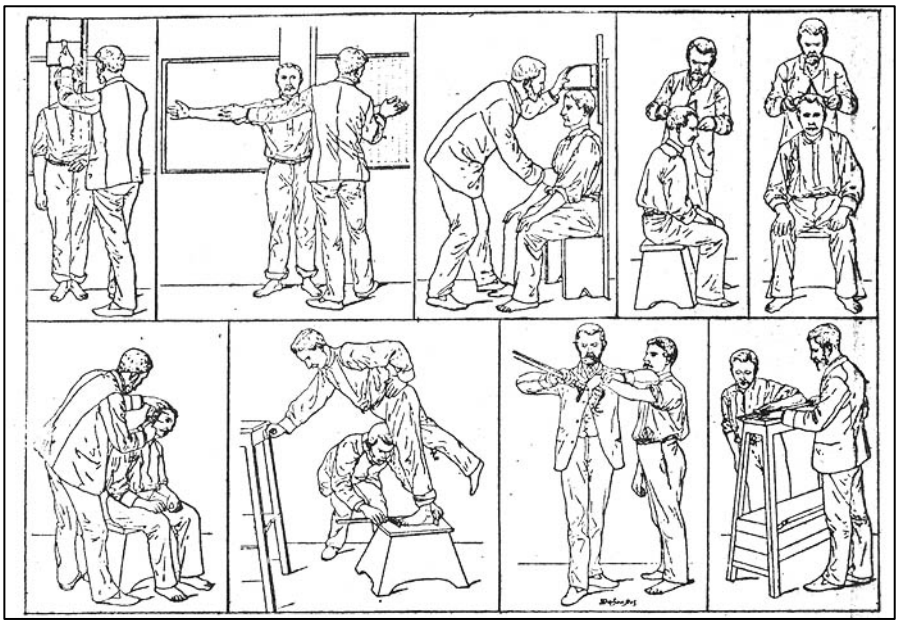

Figura 1 [9]. Pagina del libro "Identification Anthropométrique" de Bertillon en el que explica su método

La efectividad del sistema se puso en duda al presentarse problemas con el uso de

distintos sistemas de unidades para las medidas y sobre todo por la dificultad en la distinción de sujetos extremadamente parecidos como los gemelos. A raíz de estos problemas se adoptó la huella digital como nuevo método para la identificación de sospechosos[10].

Tal fue el éxito de este método creado por sir Francis Galton y mejorado por Juan Vucetich en 1982 que a día de hoy sigue siendo el método más utilizado mundialmente. A pesar de esto, a lo largo del siglo XX se ha desarrollado una gran cantidad de otras modalidades biométricas. El progreso de la tecnología permite el desarrollo de modalidades nuevas, como la identificación por el patrón del iris o el reconocimiento facial (desarrollado entre los finales de la década de los 80 y principio de los 90 del siglo XX). También se han propuesto y están en desarrollo modalidades de identificación que emplean la vascularización, la firma, la voz o la manera de caminar.

2.2 Modalidades Biométricas

Es posible utilizar prácticamente cualquier característica biológica o de comportamiento de una persona para la identificación de ésta. Las características de comportamiento técnicamente pueden llegar a ser imitadas por un usuario fraudulento, por eso las características biológicas y físicas se consideran más seguras. Los parámetros que se tienen en cuenta para decir que un rasgo biológico puede emplearse como modalidad biométrica:

- La universalidad, cualquier persona debe disponer de las características extraídas, por ejemplo manos, dedos, ojos, etc.
- La unicidad, es importante que la probabilidad de encontrarse dos personas con las mismas características sea muy baja, si no nula.
- La facilidad de captura y el coste, la captura de la característica no puede suponer una molestia para el usuario ni una inversión excesiva de tiempo o dinero.
- La aceptación por los usuarios, los usuarios no deben sentirse incomodos durante en proceso de captura de la característica o la identificación por ningún motivo ya sea personal, religioso o de cualquier otra índole.
- La estabilidad de la muestra, es importante que la característica no se altere mucho con el paso del tiempo.
- La robustez frente al fraude, la característica no debe ser fácil de imitar.

A continuación se presentan algunas de las modalidades biométricas más utilizadas y/o estudiadas actualmente. Hay que comprender que cada modalidad tiene sus ámbitos de uso.

Es posible que una modalidad resulte inadecuada para alguno de estos ámbitos y es por ello que, siempre se debe hacer un estudio del entorno antes de decantarse por alguna de ellas.

- **Huella dactilar.** Es la modalidad biométrica más extendida y más estudiada. En occidente se utiliza desde el siglo XIX y existen numerosos estudios que prueban la estabilidad de la huella dactilar a lo largo del tiempo así como la unicidad de las muestras. A pesar de que actualmente existen modalidades más exactas, como por ejemplo la identificación por iris, sigue siendo la modalidad preferida de los cuerpos policiales en todo el mundo por su bajo coste y la facilidad de extracción de las características. El uso de esta modalidad por los cuerpos de seguridad del estado ha hecho que socialmente se la asocie con actos delictivos, a veces provocando rechazo en la población.

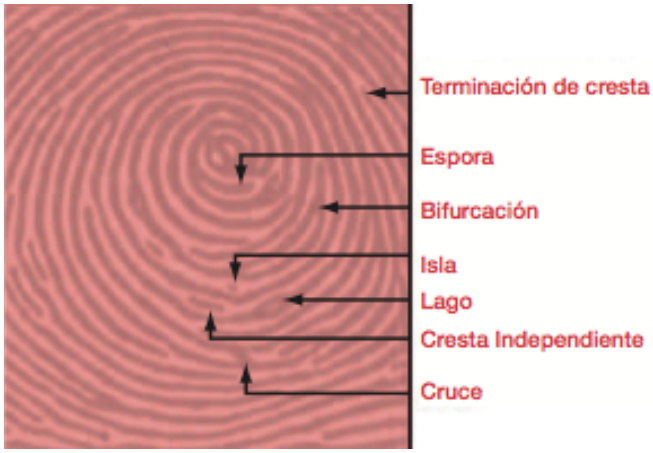

Figura 2 [11]. Minucias, detalles de la huella en las que se basa la identificación por huella dactilar

- **Voz.** Actualmente en proceso de estudio y desarrollo. Resulta muy atractiva la idea de la identificación de manera remota, sin contacto físico o incluso a distancia, valiéndose de dispositivos de telecomunicación por voz. Sin embargo, esta modalidad presenta una baja estabilidad ya que la voz es sensible a variaciones tan comunes como el estado del ánimo, salud o la edad del usuario. El umbral del sistema debe ser reducido a fin de que se identifique correctamente a los usuarios haciendo esto la modalidad poco segura.

- **Iris**. Anillo que rodea la pupila y por el cual diferenciamos el color de ojos de la personas. Los estudios demuestran que la unicidad del iris es muy superior a la de la huella dactilar siendo las muestras, por lo menos, tan estables como lo son las de las huellas dactilares. La extracción de las muestras es relativamente sencilla y la modalidad no acarrea las connotaciones policiales. Sin embargo, la dificultad de adquisición de imágenes con la calidad necesaria, para sus algoritmos, hace que no sea tan extendida como la huella dactilar. Al no tener aplicaciones forenses (no queda rastro tras el uso de esta modalidad), tampoco dispone del apoyo de este sector.

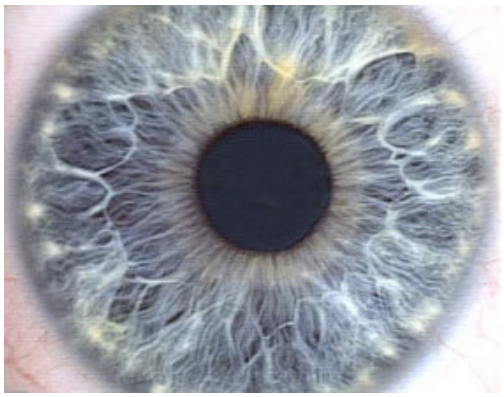

Figura 3. Es posible distinguir hasta 200 características diferenciadoras en un iris humano por lo que se posiciona como una de las modalidades más seguras que existen

- **Retina.** Esta es otra modalidad que utiliza el ojo humano; en este caso la geometría vascular de la retina proporciona la información de la identidad. Esta es una de las modalidades más seguras, superando en unicidad incluso a la del iris. Consiste en el escaneo del fondo de la retina por medio de un láser, algo a lo que la mayoría de la población siente rechazo y que junto al elevado precio de los equipos hace que sea exclusiva de entornos de altísima seguridad.

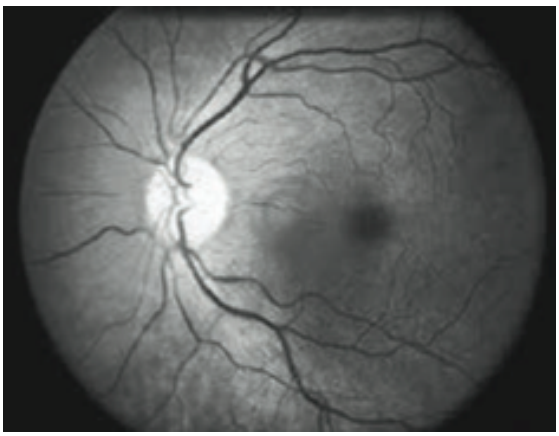

Figura 4. El patrón venoso de la retina es único por lo que esta modalidad presenta unas tasas de falso positivo ínfimas, superadas únicamente por el ADN

- **ADN.** Es la modalidad con mayor tasa de éxito, próximo al 100% con la única excepción del caso de gemelos idénticos[12]. Las técnicas de extracción (saliva, sangre, etc.) resultan muy invasivas para el usuario por lo que la modalidad tiende a ser rechazada. El coste de procesado de las muestras es muy superior al de cualquier otra modalidad como contrapartida a su bajísima tasa de fallo. Además, la modalidad basada en ADN requiere un tiempo de procesado que impide que sea utilizada en sistemas con requisitos temporales.
- **Cara.** Otra modalidad que está en pleno crecimiento y con espectaculares avances, especialmente en la primera década del siglo XXI; su desarrollo comenzó en los años 90 del siglo XX [13]. El uso de cámaras estereoscópicas que permiten la reconstrucción de modelos 3D de las caras, ha hecho que los algoritmos de esta modalidad sean 100 veces más eficaces que en los años 90. El problema de esta modalidad es que presenta muy poca tolerancia a los cambios ambientales con los que se realiza la toma de muestras, especialmente a los cambios de iluminación. Además, esta modalidad presenta poca estabilidad, existe baja tolerancia con cambios que se consideran habituales, tales cambios de expresión, vello facial o las gafas.
- **Geometría de la mano.** Esta es la modalidad biométrica más rápida puesto que no tiene en cuenta tantas características como otras. Con pocas características para procesar, la unicidad que presenta la modalidad es limitada. Por ello, se recomienda su uso en entornos con pocos usuarios pero que precisen velocidad en la verificación.

Figura 5. Se extraen características como curvas de los dedos, grosor y longitud, altura y anchura del dorso de la mano, etc. Todas ellas características más o menos constantes; no se tienen en cuenta detalles superficiales como cicatrices, uñas o huellas dactilares

- **Vascular.** Modalidad objeto de estudio de este TFG. El patrón biométrico es extraído de la geometría del árbol de venas de la mano, un dedo o de las muñecas. Al

tratarse de venas estamos ante un patrón biométrico interno con lo que no deja rastro y su extracción solo es posible en presencia del individuo eliminándose las connotaciones policiales de la huella dactilar, a la vez que la posibilidad de su uso en el ámbito forense. La estabilidad de las muestras es alta y aunque no hay estudios a gran escala sobre su unicidad, se estima que sea muy alta[14]. Al igual que con la huella dactilar, la formación de los capilares es aleatoria.

Con las características presentadas esta modalidad es indicada para entornos de alta seguridad donde no es posible el uso de la huella dactilar ya sea debido a las características del entorno o los requisitos de seguridad impuestos.

Figura 6. Presentando una unicidad por lo menos tan alta como la de la huella dactilar, no tiene las connotaciones policiales de esta y tratándose de un patrón biométrico interno, no es posible su uso sin la presencia del individuo

- **Firma.** Modalidad biométrica conductual, probablemente la más antigua de todas. Con la técnica clásica de identificación por firma, con el entrenamiento adecuado era posible la falsificación de la muestra por un usuario no genuino existiendo auténticos expertos en este campo. En las técnicas modernas a parte del aspecto visual de una firma se tienen en cuenta otras características de ésta para verificar al usuario que hacen la falsificación extremadamente complicada. Las nuevas características que se tienen en cuenta son el tiempo en el que se realiza la firma, paradas, la presión sobre el papel, la inclinación del bolígrafo, etc.
- **Dinámica del tecleo.** Debido a que cada usuario interacciona con el teclado de una manera diferente, es posible la identificación de una persona por medio de su ritmo de tecleo. Este resulta un método biométrico conductual de baja unicidad pero aun así es un posible sustituto a las contraseñas para el acceso a aplicaciones on-line.

- **Forma de andar.** Esta modalidad está en pleno desarrollo. Todas las personas caminan de forma distinta, debido a su constitución física y costumbres personales y es por tanto una característica con la que es posible la identificación de usuarios. La modalidad es muy dependiente de las condiciones del individuo en le momento de

toma de muestras , influyen factores tan diversos como calzado o el cansancio. Es posible su uso en la identificación de sujetos ya sea en grabaciones de video o en tiempo real.

2.3 Sistemas Biométricos

Todos los sistemas biométricos siguen un esquema predefinido de obtención y verificación de datos. El esquema es independiente de la modalidad biométrica utilizada y se compone de los siguientes dos etapas.

- **Reclutamiento.** En esta etapa se toman una o varias muestras del usuario, se procesan y se recopilan su características para extraer el patrón biométrico del usuario, finalmente el patrón se guarda en la base de datos para su posterior uso. Es importante saber que esta etapa se suele realizar bajo la supervisión de una persona encargada de asegurar el correcto procedimiento y la veracidad de la identidad del usuario. Esta etapa se realiza una sola vez.
- **Verificación.** Etapa que se puede realizar las veces que sea necesario para identificar al sujeto. Para el usuario, la experiencia de la verificación es muy similar a la del reclutamiento, el usuario ofrece, para la adquisición, la característica propia del método que se esté usando (dedo para huella dactilar, ojo para iris, etc.) e internamente el sistema repite varios de los pasos que ya se hicieron previamente en el reclutamiento. La nueva muestra se procesa y se vuelven a extraer las características de esta, después, las características de la nueva muestra se comparan a las del patrón guardado en la base de datos. Tras esta comparación, el sistema devuelve el resultado de la identidad o no del usuario.

Es importante entender que la toma de decisiones del sistema sobre si la identidad del usuario, no es exacta. Una muestra tomada en el momento de la verificación nunca va a ser 100% exacta al patrón almacenado. Debido a que dos muestras nunca serán iguales por deterioro de la característica física, cambios en la conducta del usuario o cambios en el ambiente o los instrumentos de captura, se hace necesario establecer un umbral de semejanza entre la muestra verificada y el patrón. De esta manera se consigue verificar la autenticidad de los usuarios en función la puntuación obtenida en la comparación de la muestra y el patrón. Así, con un umbral del 60% toda muestra con una semejanza de 60% o más sería dada por la del usuario verificado y una muestra con una semejanza menor al 60% sería considerada como no correspondiente a ese usuario. En la Figura 7 se aprecia el esquema del sistema biométrico descrito.

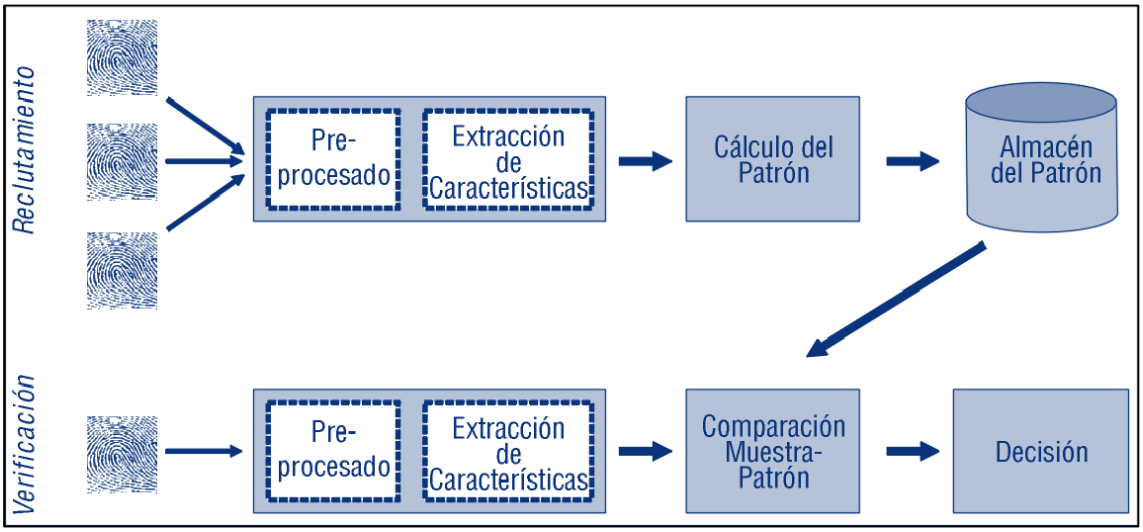

Figura 7 [15]. Esquema de funcionamiento de un sistema biométrico.

Como ya se ha indicado, la verificación es una comparación de las características de una muestra con las de un patrón del mismo usuario, es decir, es una comparación 1:1 para confirmar la identidad de un usuario. Existe otra posible funcionalidad, identificación, en la que la comparación de las características de la muestra se lleva a cabo con más de un patrón. En esta funcionalidad se desconoce a priori la identidad del usuario y lo que se busca es descubrirla buscado la mayor semejanza entre la muestra de este con todos los patrones de la base de datos. Por la particularidad de este modo resulta imposible emitir un veredicto de verdadero o falso por lo que los resultados emitidos por el sistema son las puntuaciones de semejanza que se han obtenido durante el proceso de comparación y le corresponde a otro sistema o un operario tomar la decisión sobre estos resultados.

2.4 Estandarización en la Biometría

Dentro del campo de la informática, que es donde en última instancia se encuentra la Biometría, un estándar es una serie de normas de obligado cumplimiento que facilitan la colaboración entre distintos desarrolladores a fin de crear un mercado de productos compatibles entre sí.

Así hasta que en los primeros años del siglo XXI no se empiezan a crear diversos estándares, la industria de la biometría carecía de cualquier criterio que facilitara la colaboración entre los fabricantes. La compatibilidad entre los productos de estos fabricantes era tan baja que, un comparador que adquiriera unos terminales de lector de huellas a una empresa no podía hacer otra cosa que adquirir el resto del sistema a la misma empresa, puesto que no tenía ninguna garantía de que estos terminales funcionaran correctamente con otras partes del sistema de un segundo fabricante. Esto complicaba tanto el desarrollo de nuevos productos como la libre competencia y también impedía a muchos potenciales compradores adquirir estos productos debido a su elevado precio.

En 2001 el ANSI (American National Standards Institute) estableció el estándar ANSI X.9.84 definiendo las condiciones de desarrollo de los sistemas biométricos en la industria de servicios financieros centrándose en lo referente a la seguridad de la transmisión y el almacenamiento de datos. Al año siguiente, ANSI en colaboración con el consorcio BioAPI, presentaron el estándar ANSI/INCITS 358 que garantizaba la compatibilidad entre sistemas. Finalmente en 2003 la estandarización se hacía internacional con el comité ISO/IEC JTC1/SC37 que se encarga de estandarizar todos los aspectos de la industria del reconocimiento biométrico tales como terminología, métodos de evaluación, interfaces de programación y los tipos de datos permitidos en esta.

Es precisamente este comité el que está desarrollando la norma ISO/IEC 30106-3 que implementa el estándar BioAPI en los lenguajes programación con orientación a objetos como Java o C# y es al estándar de este último lenguaje al que se han adaptado los algoritmos en el presente TFG. Se profundizara en el proceso de la adaptación del código al estándar en el apartado 5.4.

Es importante señalar que la norma está en continuo proceso de mejora, igual que el resto de normas desarrolladas por el comité. Tanto que a día de hoy no existe aún versión definitiva[16]. Actualmente la estandarización dentro del campo de la biometría sigue teniendo mucho camino por recorrer hasta ponerse a la altura de otros campos como, por ejemplo, la electrónica.

3 Tecnologías Asociadas

En este capítulo se comentan brevemente las herramientas usadas durante el desarrollo de este TFG. Primero se hablara sobre el lenguaje de programación elegido (C#) y las librerías de funciones de tratamiento de imagen que se utilizaron (Emgu CV). Después, se explicará la importancia y los detalles técnicos del estándar BioAPI uniéndolo directamente con el apartado 2.4 del capítulo anterior.

3.1 Lenguaje C#

C# es un lenguaje de programación, creado en los primeros años del siglo XXI, orientado a objetos, que se diseñó pretendiendo unificar la facilidad de uso de Visual Basic y la potencia de C++. Como todo lenguaje de programación orientada a objetos, se basa en los conceptos de herencia, abstracción, polimorfismo, encapsulamiento, etc.

El lenguaje, originalmente basado en C++, se diseñó para ser más cómodo para el programador por lo que se eliminaron algunas característica como los punteros, la herencia múltiple, ficheros de cabecera, etc. y se añadieron otras, para facilitar su uso, como seguridad de tipos o la recolección automática de basura.

Similar a otros lenguajes que ya lo implementaron antes (como por ejemplo Java), C# está pensado para ejecutarse por medio de una máquina virtual que provee el CLR (Common Language Runtime).

3.1.1 Similitud con otros Lenguajes C Estudiados

Durante el desarrollo del Grado en Electrónica, los alumnos se familiarizan con los lenguajes típicos de programación como C y C++ lo cual ha sido muy útil en el desarrollo de este TFG puesto que C# se ha desarrollado pensando en una fácil aceptación por los programadores habituales de C y C++.

Así, un usuario con conocimiento básico del funcionamiento de C/C++ encuentra grandes similitudes con C#, comenzando con la misma estructuración básica del código (clases, métodos, objetos, etc.) y terminando en la sintaxis, gran parte de la cual se conserva de C/C++.

Existen por otro lado bastantes diferencias, todas sencillas de asimilar y en su mayor parte facilitan el trabajo del programador, estas son lagunas de ellas:

- No se utilizan los ficheros de cabecera. Esto dinamiza la programación, ya no es necesaria la doble declaración de las variables y las funciones.

- La gestión de memoria es automática. Esto resulta muy cómodo para el programador ya que no tiene que preocuparse de destruir los objetos que ya no se van a usar puesto que se hace automáticamente.
- No se usan los punteros. El gran quebradero de cabeza de todos los usuarios principiantes de los lenguajes orientados a objetos se ha eliminado del uso común de C# y en su lugar se utilizan las referencias a objetos.
- La declaración del método *main* es distinta.
- Las sentencias *switch* y *foreach* han sufrido ciertos cambios, ampliando su funcionalidad.

Existen muchas más diferencias entre los lenguajes en cuestión aunque los comentados son los que se han encontrado durante el desarrollo del TFG.

Por último, es importante señalar la principal característica diferenciadora de C# con respecto a C/C++ y es el apartado gráfico. La parte visual de C#, con la que se ha construido la aplicación en la que se basa la memoria de este TFG, es con diferencia mucho más fácil e intuitiva de utilizar que la que existe en el Visual C++. Facilita enormemente la creación de las típicas aplicaciones Windows, de botones y menús desplegables, facilitando por lo tanto el trabajo para el programador y ayudando a crear un producto atractivo para el usuario. Ver apartado 5.1.2.1.

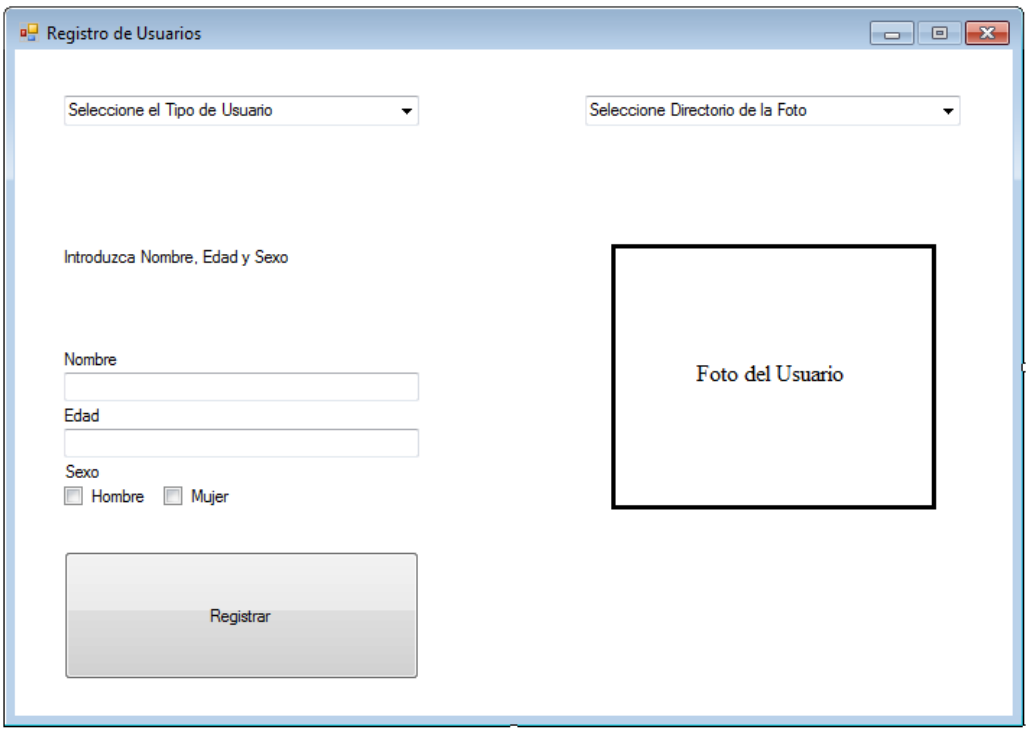

Figura 8. Ventana de una típica aplicación creada con C# que incluye menús desplegables, cajas de texto, botones, etc.

3.1.2 Librerías de código abierto: Emgu CV

Puesto que el cuerpo del TFG consistía en el tratamiento de las imágenes, eran necesarias unas librerías que proveyeran las funciones capaces de realizar los cambios necesarios en las imágenes. Cierto es que una instalación básica del programa C# ya incluye numerosas librerías de este tipo, capaces de realizar la mayoría de los cambios que son necesarios, pero el trabajo debía ser realizado usando las bibliotecas de código libre. Al ser libre, de ser necesario era posible el acceso al código y su modificación, cosa impensable en las librerías propias de C#. Además, en su última versión (Emgu.CV-2.4.9 Beta[17], la utilizada en el TFG) ya se incluyen todas las funciones necesarias para este tipo de trabajo (se comentan las excepciones en el capítulo 5.

Emgu CV es un envoltorio .NET que permite el funcionamiento entre plataformas de las librerías de tratamiento de imágenes OpenCV. Hace posible que las funciones de OpenCV sean llamadas desde los lenguajes compatibles con .NET tales como, C# Visual Basic, Visual C++, etc. Con los ajustes necesarios el envoltorio puede ser utilizado tanto en Windows, Linux y dispositivos Android como en Mac OS X, iPhone y iPad.

Ya que Emgu CV es solo un envoltorio, en realidad lo que se utiliza son las bibliotecas de Open CV y estas existen oficialmente desde 1999 bajo la licencia BSD, libres tanto para uso comercial como en investigación. Estas bibliotecas, programadas y optimizadas en lenguajes C/C++ fueron diseñadas inicialmente para avanzar en la investigación, difundir el conocimiento y mejorar los usos comerciales de la visión artificial. Se proporcionó código abierto en una infraestructura común, facilitando el intercambio de código entre los desarrolladores, y todo ello bajo una licencia gratuita[18]. Sus aplicaciones van desde tratamientos de imágenes en 2 y 3D hasta la detección de movimiento y reconocimiento facial.

3.2 Estándar BioAPI

BioAPI es un estándar de aplicación biométrica en el que se definen dos interfaces, de programación de aplicaciones o API (Aplication Programming Interface) y de proveedores de servicios o SPI (Service Provider Interface). Por medio de estas es posible la comunicación de dispositivos de distintos fabricantes persiguiéndose con ello el desarrollo de sistemas de seguridad más sólidos y una mayor competencia entre los fabricantes como ya se explicó en el apartado 2.4.

3.2.1 Historia

Durante la conferencia CardTech/SecureTech de 1998, con el apoyo de gigantes de la industria informática como BMP y HP, se crea el consorcio BioAPI. El objetivo de este consorcio inicialmente era la creación de un estándar biométrico que ofreciera una alternativa a otras propuestas de estandarización que ya estaban surgiendo, incluyendo propuestas tan

interesantes como, la creación de una API independiente del paramento biométrico, del hardware e incluso del sistema operativo, sobre el que funcionase la aplicación[19].

Dos años más tarde apareció la primera especificación del estándar que ya cumplía con sus objetivos básicos y cubría aspectos como el reclutamiento, verificación, identificación de usuarios, captura y proceso de datos, etc. y todo ello a la vez que era compatible con los estándares ya existentes en ese momento como el Human Authentication API.

Figura 9. Logo adoptado por el consorcio BioAPI

3.2.2 Arquitectura, BioAPI C#

La arquitectura de BioAPI permite un rápido desarrollo de aplicaciones biométricas facilitando una estructura predeterminada al programador evitándole así crear una propia. Desde un punto de vista generalista, BioAPI ofrece interfaces sencillas para la creación de las aplicaciones y métodos estandarizados tanto de acceso a funciones, algoritmos y dispositivos como de gestión y almacenamiento de datos obtenidos. Se estandariza el modo en el que las aplicaciones se comunican con los dispositivos y la forma en la que los datos son gestionados y almacenados pero sin entrometerse en las características diferenciadoras de cada modalidad ni en la manera en la que estas generan sus datos.

Un sistema biométrico completo, con el estándar BioAPI implementado se estructura en capas, según la arquitectura API/SPI, ver Figura 10.

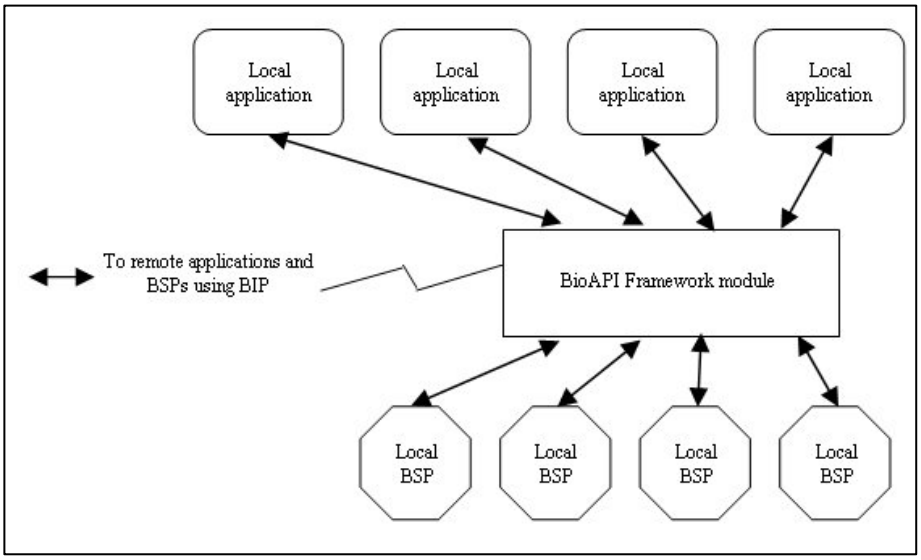

Figura 10 [20]. Arquitectura del estándar BioAPI

En el primer nivel (nivel bajo) encontramos los Proveedores de Servicios Biométricos o BSP (Biometric Service Provider) que ofrecen sus servicios a los dispositivos biométricos finales (escáneres, cámaras y sensores de todo tipo) que técnicamente están un nivel más abajo, y a los proveedores de algoritmos, gestores de datos, etc. El segundo nivel es la capa del Framework, separa las aplicaciones locales del resto del programa haciendo que las aplicaciones no "sepan" que funciones o algoritmos se están usando para proveerlas de los datos que maneja. En el nivel superior se encuentran las aplicaciones locales, con las que interactúa el usuario. Estas utilizan funciones definidas por el estándar que después, a través del Framework, usan las funciones propias del algoritmo que se esté usando, ubicado en los BSPs. Las flechas representan las interfaces API y SPI encargadas de la comunicación entre el Framework, las aplicaciones y los BSPs.

La arquitectura explicada, originalmente aplicada a la versión del lenguaje C de BioAPI, se ha mantenido integra en la adaptación al lenguaje C# ajustándose al ISO/IEC 19784-1, con lo que también es aplicable a la aplicación desarrollada para este TFG.

3.2.2.1 Estandarización en el Reconocimiento Vascular

Los datos obtenidos durante el reconocimiento vascular no son ninguna excepción para la capacidad característica del estándar de tratar por igual los datos biométricos, provengan de la modalidad biométrica que provengan. Todos los datos biométricos, sean de la modalidad biométrica que sean, realizan los "viajes" entre las distintas capas del estándar encapsulados en la estructura de datos BIR (Biometric Identification Record). Dentro de esta estructura el dato biométrico debe estar convertido al tipo *array* por lo que en el fondo el estándar no "sabe" con que tipo de dato biométrico trata, únicamente la aplicación dispone de esa información.

Referente a la transmisión de información relacionada con imágenes de patrones vasculares humanos, existe normativa ISO, concretamente la normativa ISO/IEC 19794-9:2007.

Esta normativa, diseñada para aplicaciones que requieren intercambio de muestras crudas (sin procesar) o procesadas, define las condiciones que debe cumplir toda muestra biométrica, vascular para su almacenamiento o transmisión. Estas condiciones se cumplen en la norma BioAPI y por lo tanto en este trabajo se cumple con la normativa mencionada.

4 Diseño de la Solución

En este capítulo se va a analizar el objetivo del TFG y las vías que se han tomado para el diseño de la solución, partiendo de la idea inicial hasta explicar la versión final de la solución propuesta.

4.1 Planteamiento del Problema

El objetivo de este trabajo es el portado de un algoritmo de identificación vascular desde el lenguaje M (MATLAB) al lenguaje C#. Este algoritmo debe funcionar en una aplicación específicamente creada para esta comprobación y esta a su vez deberá cumplir con el estándar BioAPI.

El principal problema que se presenta es el portado del código del algoritmo. No existe ningún método concreto de adaptación de un lenguaje de programación a otro. Tratándose de dos lenguajes de programación bastante distintos, el proceso habitual es reescribir el código original manualmente en el lenguaje de destino. Se tratará este tema más a fondo en el siguiente apartado.

Por otro lado existe la necesidad de crear una aplicación visual con la que el usuario pueda interactuar sin dificultad y es en esta aplicación donde el código portado se debe aplicar, a los datos biométricos. En un sistema biométrico completo, que es lo que simula ser la aplicación creada, deben existir distintas funcionalidades (reclutamiento, verificación, identificación, etc.) que en muchas ocasiones realizan operaciones similares con las muestras biométricas, de manera que conseguir reutilizar el código en estas funciones es un objetivo a alcanzar con el fin de conseguir un programa eficiente.

Por último, el tercer punto importante del trabajo es la inclusión del estándar BioAPI en la aplicación. Para tal fin existen dos caminos.

- Primero, tomando a modo de plantilla el estándar, que ya existe en lenguaje C# bajo la norma 30106-3 Object Oriented BioAPI, ir añadiendo las funciones del algoritmo en los correspondientes BSPs y BFPs a la vez que las vamos “traduciendo” de un lenguaje a otro. Esta parece la manera óptima ya que de esta manera es posible ir resolviendo los errores según van apareciendo. La plantilla del estándar ha sido suministrada al alumno por el grupo de investigación (GUTI), en el que se ha realizado el TFG.
- Segundo, crear y optimizar la aplicación con el código portado, sin ocuparnos del estándar hasta que la aplicación funcione completa y correctamente. Después, adaptarla al estándar y volver a depurar el código completo por si aparecen errores. Se debaten ambos caminos y se explica el porqué de la elección en el siguiente apartado.

4.2 Planteamiento de la Solución

Antes de comenzar con el desarrollo del trabajo es necesario decidir y plantear adecuadamente la solución a los problemas que se han planteado en el apartado anterior. Se van a explicar en este apartado las decisiones tomadas para abordar los problemas y se describirán brevemente las soluciones.

4.2.1 Código

Al igual que ocurre en la traducción de un texto convencional, el proceso depende del lenguaje original y el lenguaje destino. En la programación, cuando dos lenguajes cumplen con el mismo paradigma (programación orientada a objetos, programación funcional, programación lógica, etc.) la traducción entre ambos suele ser más sencilla. Se debe repetir la misma estructura del código solo que cambiando la sintaxis. En muchas ocasiones incluso existen servicios de traducción automática aunque la detección de posibles errores en la traducción es complicada, por eso los programadores prefieren no utilizar estos servicios.

En la traducción entre dos tipos distintos de lenguajes se adoptan otras estrategias. En este trabajo es necesaria la traducción desde el lenguaje M que sigue el paradigma de la programación estructurada y la mayoría de sus funciones trabajan con matrices, al lenguaje C# que sigue el paradigma de programación orientada a objetos tradicional. La manera más habitual de abordar esta traducción sería la reescritura manual del código completo. Analizando los métodos y funciones utilizados en el programa original, se buscan métodos lo más parecidos posible en el lenguaje destino y de esta manera se va reconstruyendo la estructura del código desde cero. Este método no es perfecto y en muchas ocasiones se hace necesario introducir cambios en la estructura del código con tal de conseguir un resultado lo más parecido al del código original.

4.2.2 Aplicación

C# es un lenguaje de programación pensado para crear aplicaciones visuales y ofrece numerosas facilidades al programador para tal fin. Por otro lado, casi todas las herramientas que ofrece para la creación de las aplicaciones funcionan a modo de eventos, lo que posibilita el uso del algoritmo por partes. Esta capacidad de dividir el algoritmo en eventos se aprovecha para la parte de demostración del funcionamiento del algoritmo (ver apartado 5.2.1) y la reutilización de algunas de sus partes en varias funcionalidades de la aplicación.

Para simular correctamente el funcionamiento de un sistema biométrico completo, la aplicación deberá disponer de las siguientes funcionalidades básicas:

- **Reclutamiento**. Con esta funcionalidad la aplicación será capaz de inscribir en la base de datos usuarios nuevos. Una vez inscritos, los usuario podrán utilizar el sistema para identificarse y además sus patrones serán utilizados en todos los procesos de identificación que se realicen. Funciona según el siguiente flujograma.

La aplicación creada solamente simula el funcionamiento de un sistema biométrico. Al contrario de cómo seria en un sistema completo, en esta funcionalidad, en esta aplicación no se toman datos biográficos del usuario puesto que ya se "sabe" a quien pertenecen las muestras (ver apartado 5.3.3).

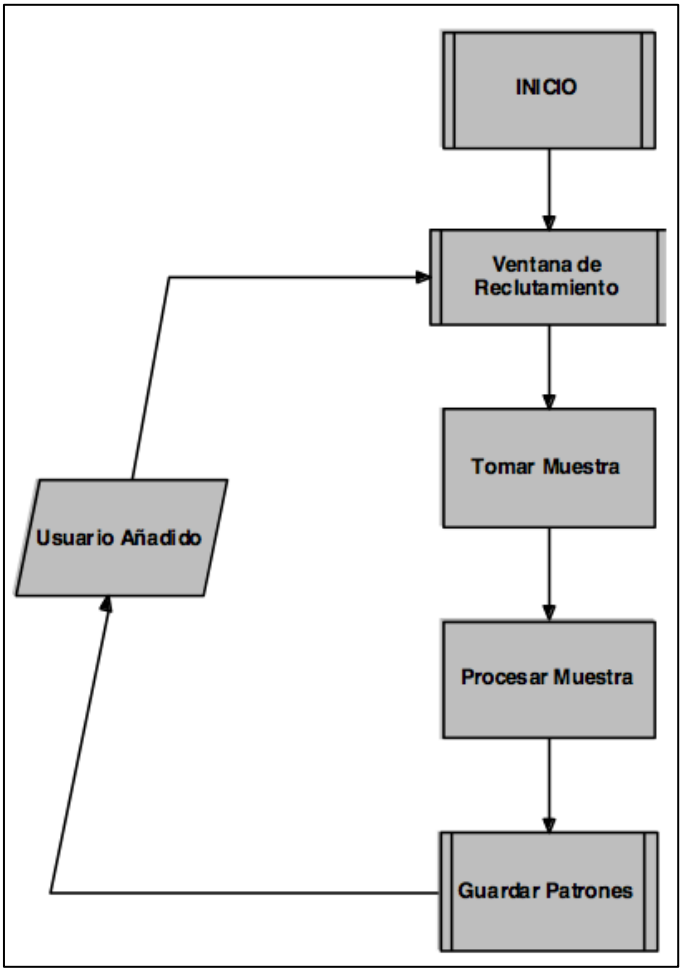

Figura 11. Flujograma de la parte de Reclutamiento de la aplicación

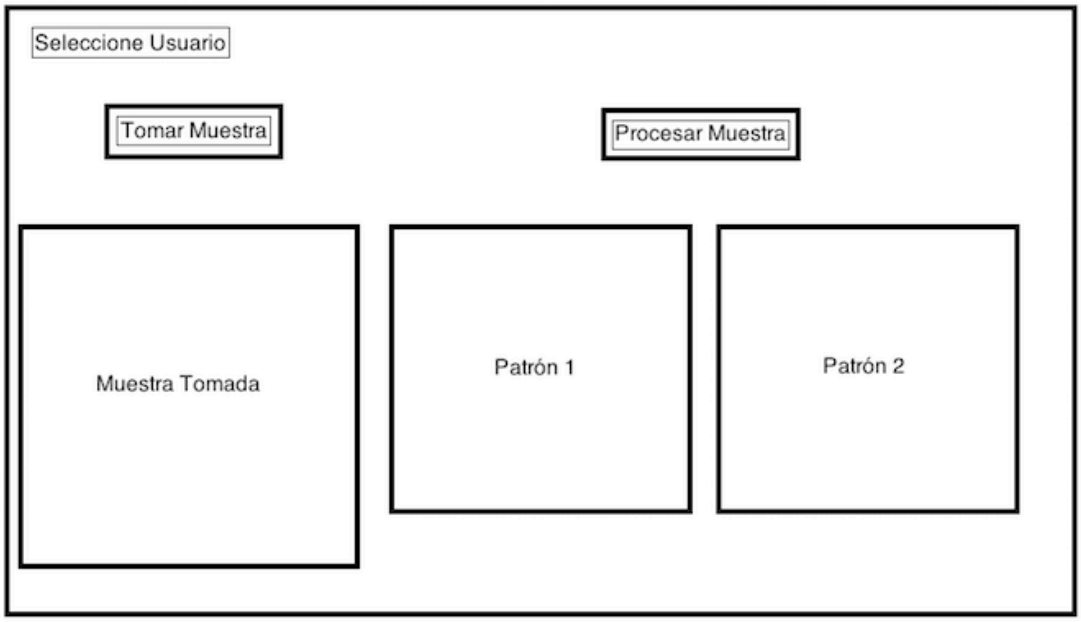

Figura 12. Boceto inicial del aspecto planteado para la parte de Reclutamiento

- **Verificación**. Funcionalidad con la que la aplicación puede comprobar si el usuario es quien dice ser. Únicamente los usuarios previamente reclutados pueden obtener una respuesta positiva. En este proceso, el usuario suministra su identidad a la aplicación al mismo tiempo que suministra una muestra biométrica. Como la aplicación ya posee el patrón biométrico del usuario, previamente reclutado, tras el tratamiento de la muestra, compara esta con el patrón almacenado en su base de datos y entonces emite el resultado de la comparación. En un sistema comercial, esta funcionalidad únicamente indica si la comparación es correcta o no, sin indicar el grado de similitud alcanzado entre muestra y patrón. Su flujograma de funcionamiento es el siguiente.

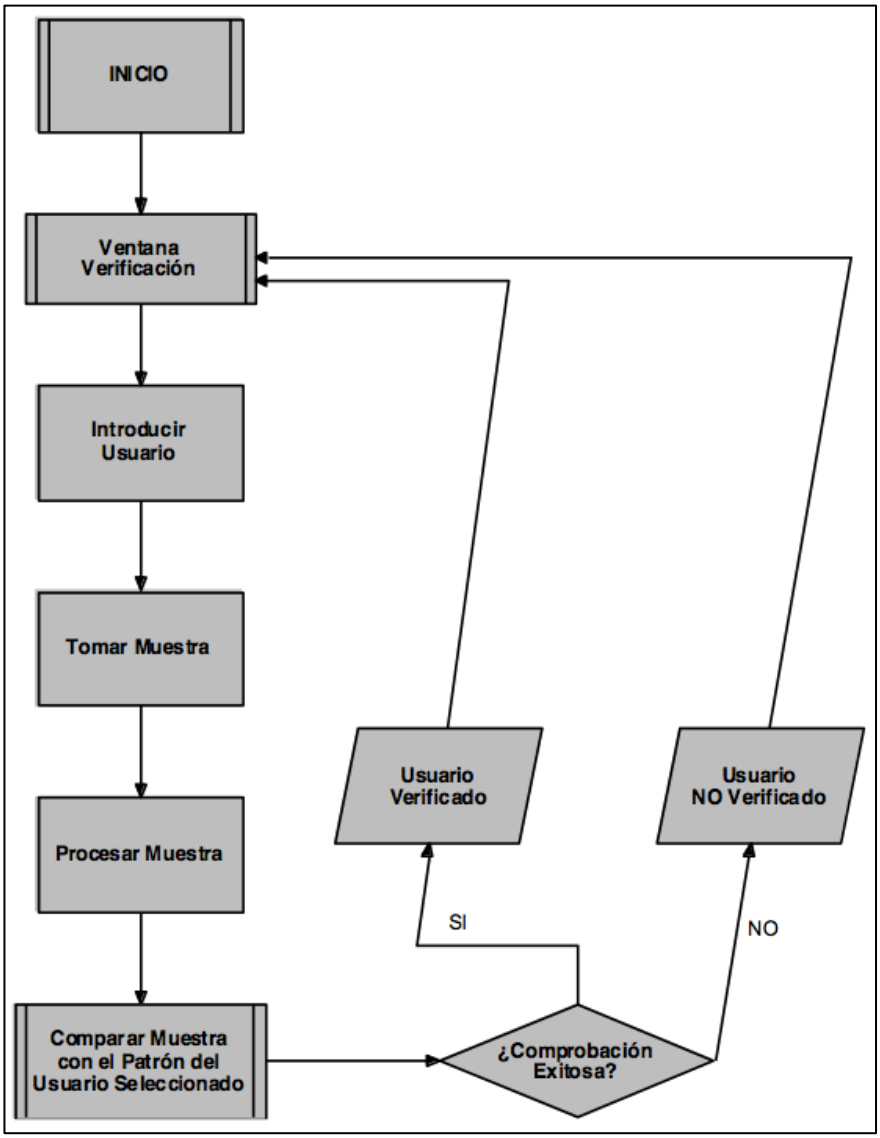

Figura 13. Flujograma de la parte de Verificación de la aplicación

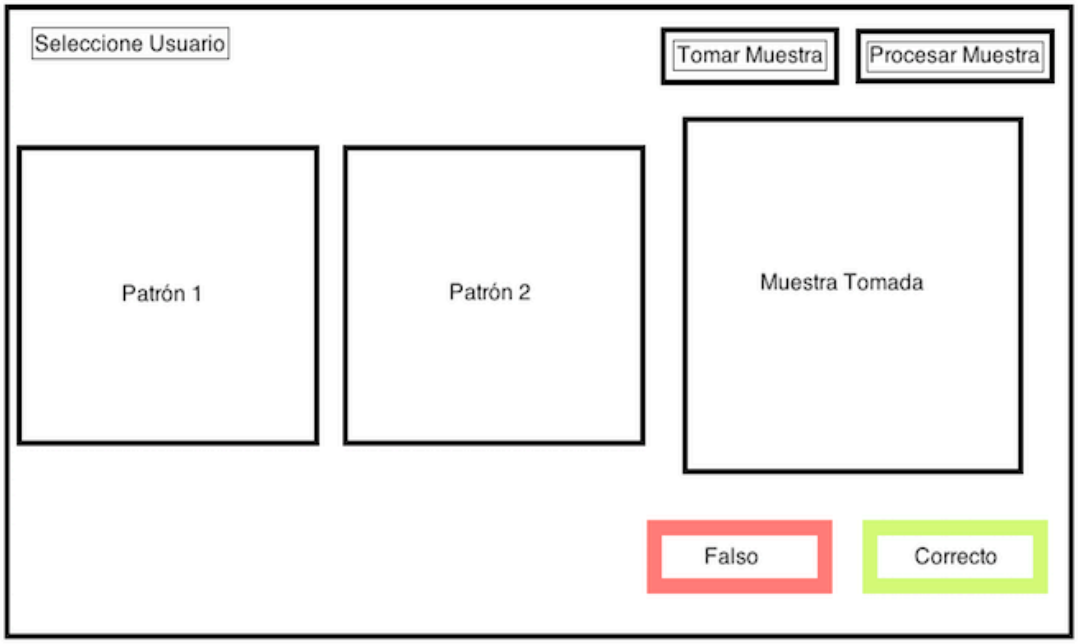

Figura 14. Boceto inicial del aspecto planteado para la parte de Verificación

- **Identificación**. Esta funcionalidad, semejante a la verificación, permite a la aplicación reconocer la identidad del usuario a partir de una muestra biométrica. Es decir, a la aplicación no se le indica quien es el usuario al que va a procesar, es la aplicación la que debe comparar la muestra que se le ofrece con toda la base de datos de patrones y devolver los porcentajes de coincidencia que ha hallado con cada uno de ellos. Posteriormente, en función del umbral establecido, se decide si alguna de las muestras presenta un índice de coincidencia lo suficientemente elevado como para considerar que es la del usuario procesado y así identificarlo. El flujograma de esta funcionalidad es el siguiente.

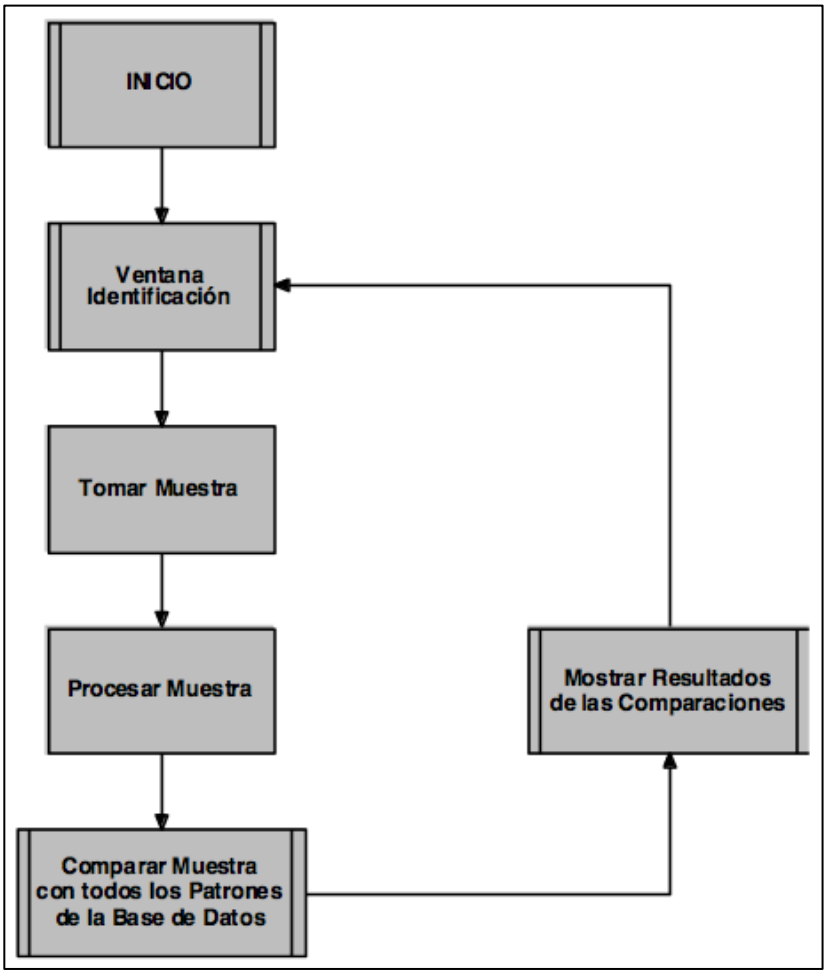

Figura 15. Flujograma de la parte de Identificación de la aplicación

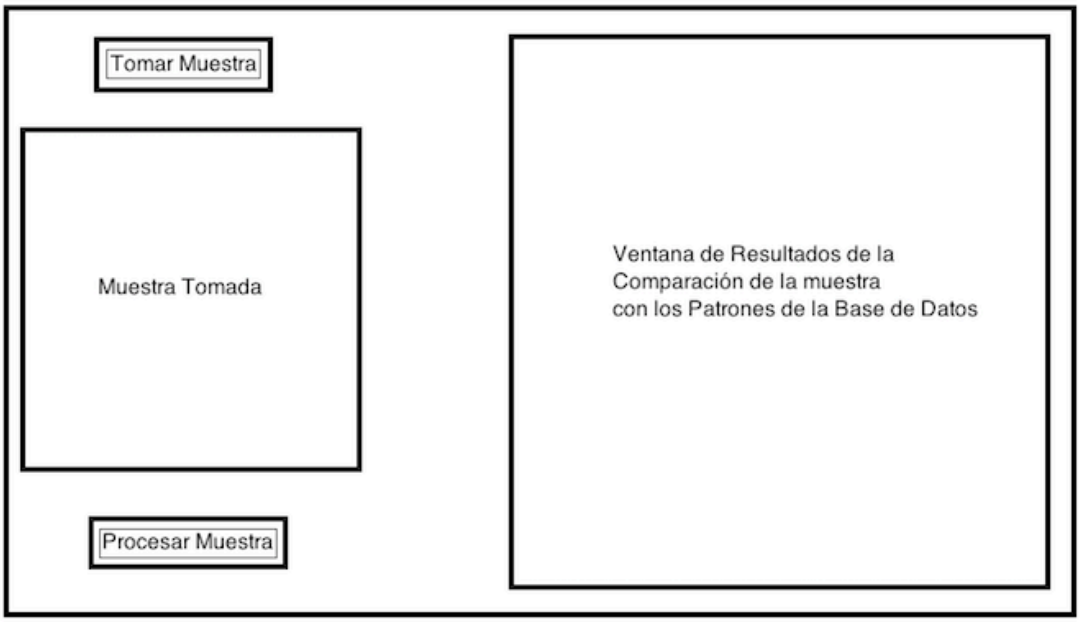

Figura 16. Boceto inicial del aspecto planteado para la parte de Identificación

4.2.3 Inclusión del Estándar

Por último, como ya se comentaba más arriba, es necesario incluir el estándar BioAPI (ISO/IEC 19784-3) en su modalidad C#, en la versión final de la aplicación. Como se describía anteriormente, existen dos posibles maneras de incluir el estándar en la aplicación. O bien hacerlo simultáneamente a la traducción del código o bien hacerlo una vez que el código ya esté totalmente traducido y funcional.

Aunque la primera opción parece el camino más lógico a seguir, y seguramente sea la opción que la mayoría de programadores experimentados habrían elegido, en la realización de este trabajo se ha optado por la segunda opción. El motivo de esta elección ha sido que al iniciar el trabajo, el alumno no tenía ningún conocimiento previo de los áreas que se abarcan en él (esto se comenta más a fondo en el apartado 5.1.2). Se consideró que aplicar el estándar, ya de por si complejo, a la vez que se está ocupado en dominar el lenguaje en el que está escrito, C#, al mismo tiempo que se trabaja con un segundo lenguaje desconocido, lenguaje M, sería demasiada carga, ralentizando considerablemente el progreso del trabajo. Lo que sería la elección de cualquier programador experimentado en el lenguaje y conocedor del estándar, resulta poco viable para un novato en ambos aspectos. Por ello, se optó por la aplicación del estándar una vez se tuviera la suficiente soltura en el lenguaje destino, soltura conseguida en la práctica de la traducción del código del algoritmo.

5 Desarrollo

A lo largo de este capítulo se resumirá el trabajo desarrollado durante la realización del TFG. El capítulo está separado en apartados que representan las fases del trabajo. En cada apartado se explicarán los objetivos de la fase, la forma en la que se ha procedido para alcanzarlos, los problemas encontrados en el proceso y la manera en la que se han solucionado.

5.1 Trabajo Previo

Antes de proceder con el trabajo propiamente dicho, era necesaria una fase de adaptación a las herramientas con las que se iba a trabajar. Se comentan en este apartado el proceso de instalación de los programas utilizados y la familiarización con estos.

5.1.1 Instalación de Programas y Bibliotecas EMGU CV

Para trabajar con el lenguaje M, el cual es propio y exclusivo de MATLAB es necesaria la instalación de este programa. El proceso de instalación de este programa es bastante simple, basta con seguir las instrucciones que van a apareciendo en pantalla y tener cuidado únicamente con la opción de la versión del sistema operativo, en el caso del alumno fue x86, 32 bits.

El programa MATLAB junto a la correspondiente licencia fueron cedidos por el tutor.

El software convencional para trabajar con el lenguaje C# es el Visual Studio y es el empleado en este TFG. Se eligió la versión 2010 de Visual Studio por una mejor compatibilidad con las bibliotecas Emgu CV, según su página web[22], y sobre todo por la existencia de un extenso numero de tutoriales de programación con C# realizados con esta versión del programa. Al igual que en MATLAB, se instaló la versión compatible con la versión x86 del sistema operativo.

El programa Visual Studio y la correspondiente licencia se han podido obtener gracias a los acuerdos que mantiene la universidad con la empresa suministradora de este software, Microsoft. Desde la página web de la iniciativa DreamSpark[23], se puede descargar gran cantidad de software, con licencia académica, que también incluye sistemas operativos. Para la realización de este trabajo fue necesaria la descarga e instalación del sistema operativo Windows 7 en el ordenador del alumno. Se ha considerado que esto no se entiende como trabajo previo de este TFG y por ello no merece mayor mención.

La instalación de las bibliotecas Emgu, fue el mayor obstáculo en esta fase del trabajo. Es posible la descarga de los archivos de instalación desde la wiki de estas bibliotecas[22], donde también se indica el proceso de instalación. Aunque completo, el proceso de instalación descrito en la wiki es complejo y al tener problemas en el proceso, se tomaron como apoyo varios video-tutoriales que están disponibles en internet.

La versión de las bibliotecas que se instaló para la realización del TFG, como ya se ha indicado antes, es la Emgu.CV-2.4.9 *Beta* (versión x86, compatible con el sistema operativo). La razón concreta de la elección de esta versión es que es la primera en la que se incluían algunas de las funciones imprescindibles para el funcionamiento del algoritmo que se iba a portar. Con esta versión era posible el uso de estas funciones (concretamente, funciones CLAHE, ver apartado 5.3) aunque por otro lado se corría el riesgo de sufrir fallos de código fuente que implica trabajar con una versión *Beta* de cualquier software. Finalmente estos fallos aparecieron aunque como se explica en el apartado 5.3.1.2 se pudieron solucionar con éxito.

5.1.2 Familiarización con los lenguajes M y C# y BioAPI

Antes de proceder a trabajar con los lenguajes, era necesaria una etapa de familiarización con estos, el alumno nunca antes había trabajado con ninguno de ellos.

5.1.2.1 C#

Para proceder con la traducción del algoritmo se hacia necesario tener cierta soltura en el manejo de la sintaxis propia del lenguaje así como de la manera de implementar funcionalidad a una aplicación (como ya se ha comentado, se hace por medio de eventos). Para ello se decidió realizar un ejercicio típico de diseño y programación, que realizan todos los estudiantes de este lenguaje, y es la creación de una calculadora simple.

Durante este ejercicio se asimilaron los conceptos básicos para el diseño de las aplicaciones como el manejo de algunas de las herramientas básicas:

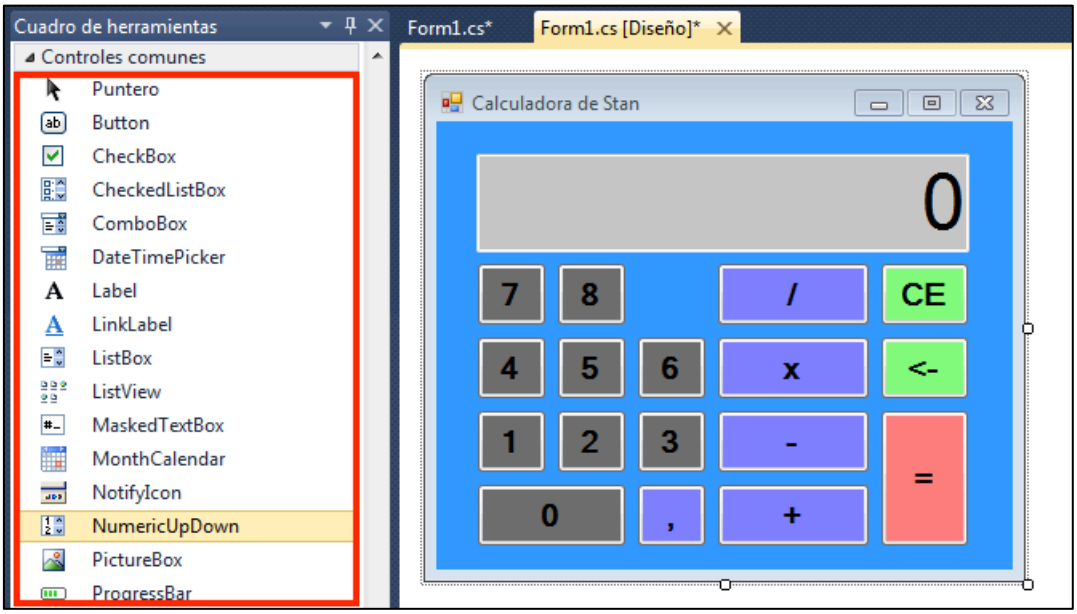

Figura 17. Herramientas comunes. Se resaltan en rojo algunas de las herramientas comunes en la construcción de ventanas con C#

Al contrario que en Visual C, donde las ventanas se diseñan de manera textual por medio de coordenadas, en C# la creación de las ventanas se hace de manera visual, arrastrando por la pantalla los botones, y demás controles, y colocándolos en el lugar deseado.

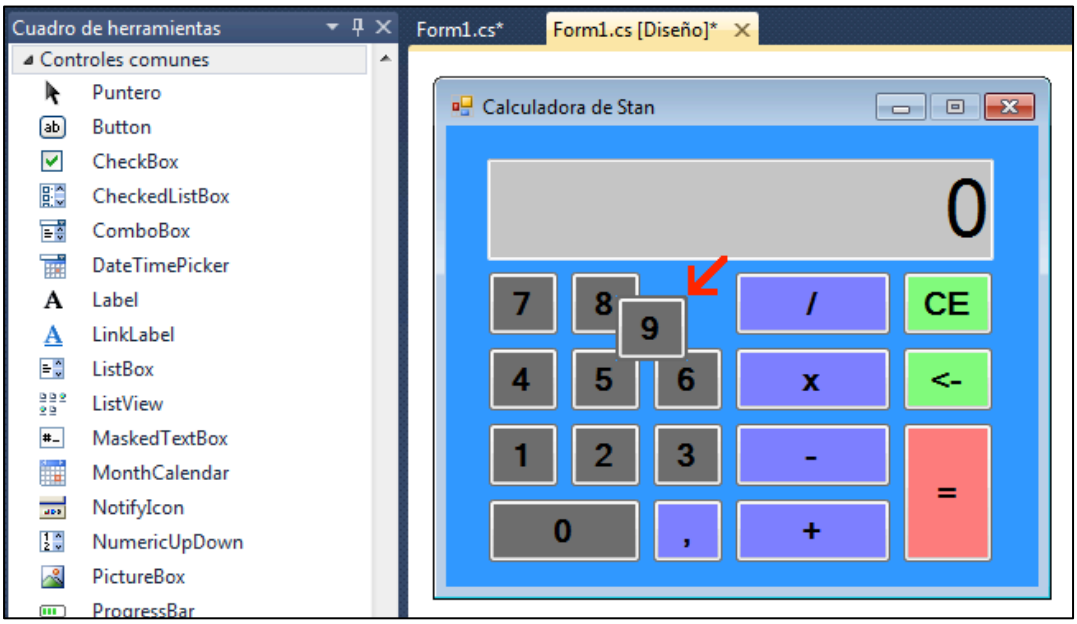

Figura 18. Construcción de las ventanas. Se resalta en rojo la colocación del botón en el lugar deseado

Después de colocar los controles de la manera en la que se desea que aparezcan en la aplicación final, se procede a su edición. Las propiedades de los controles permiten su personalización. Los posibles cambios incluyen desde simples cambios de tamaño y texto con el que aparece el control hasta numerosos cambios de color (color de fondo, de esquinas, sombras, etc.). La modificación más interesante resulta ser la posibilidad de seleccionar el tipo de evento al que queremos que reaccione el control. Por ejemplo, el evento típico de los botones es el clic (es decir, el botón realizara la acción para la que esta programado cuando se haga un clic encima de este), pero se puede cambiar para que sea un doble clic o al intento de arrastrar el botón.

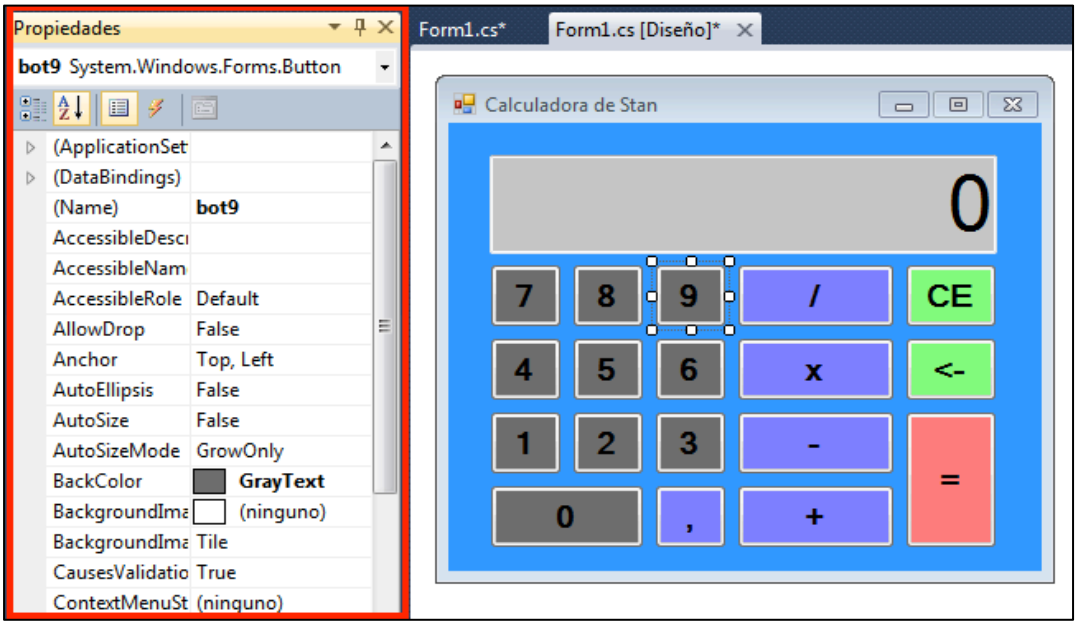

Figura 19. Edición de los controles de la aplicación. Se resaltan en rojo algunas de las opciones de las que se dispone para personalizar los controles

Por último, queda la programación de los eventos que deben realizar los controles que se hayan colocado en la aplicación. Cada control se programa por separado con el evento que se haya elegido. Si se hace doble clic sobre cualquier control desde la ventana de diseño, en la parte de código se crea automáticamente la cabecera del evento con el tipo de evento que hay por defecto (en los botones es el clic). Como se ve en la Figura 20, la sintaxis, en C#, de los eventos es muy similar a la de C++ por lo que no hubo problemas en este aspecto.

```
private void bot9_Click(object sender, EventArgs e)
{
    if (reinicio == true)
    {
        pantalla.Text = "";
        reinicio = false;
        pantalla.Text = "9";
    }
    else
    pantalla.Text = pantalla.Text + "9";
}
//Fin numeros
//Operaciones
private void suma_Click(object sender, EventArgs e)
{
    operacion = "mas";
    numero1 = double.Parse(pantalla.Text);
    reinicio = true;
    comaON = false;
}
```

Figura 20. Código de los eventos que se realizan al hacer clic en los botones de la calculadora

5.1.2.1.1 Bibliotecas Emgu CV

El uso de las bibliotecas Emgu es muy similar al de cualquier otra biblioteca disponible en C# con la instalación básica del programa, además existen numerosos video tutoriales para el uso de estas. Con el fin de asegurarse el correcto funcionamiento de las bibliotecas y ver la sintaxis de las llamadas a sus funciones se probaron algunas de estas funciones básicas.

Figura 21. Resultado del uso de la función, propia de Emgu CV, Not (invertir colores de una imagen)

5.1.2.2 Lenguaje M

El lenguaje M es muy extenso y es posible su aplicación en campos tan diversos como resolución de ecuaciones diferenciales, modelización y simulación, análisis de datos, etc. Este TFG se basa en una aplicación que utiliza el lenguaje para el procesamiento de imágenes. Esta aplicación sirvió de ejemplo en el proceso de familiarización con el lenguaje y el programa, MATLAB.

Como el trabajo del TFG se desarrolla principalmente en C# y únicamente se basa en la aplicación del lenguaje M, el aprendizaje se centro en comprender el orden de llamadas a los archivos .m (distintas componentes de la aplicación), y en saber localizar las funciones principales que constituyen el algoritmo a traducir.

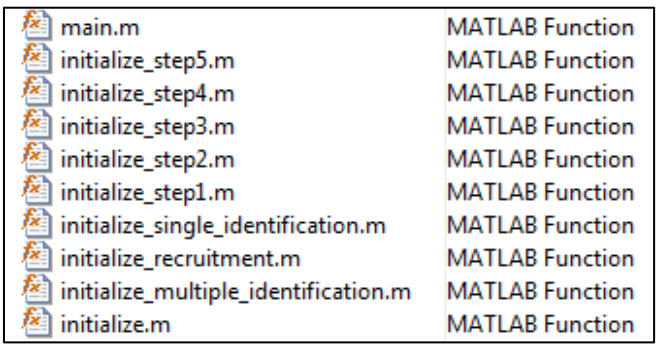

Figura 22. Algunos de los archivos .m que componen la aplicación

5.1.2.3 BioAPI (ISO/IEC 30106-3)

La familiarización con esta parte del trabajo no se produjo hasta después de acabada la traducción de todo el código del algoritmo, por lo tanto este apartado no sigue el orden cronológico, como el resto de la memoria.

Después de la traducción de todo el algoritmo y horas de trabajo con la sintaxis propia de C#, la comprensión del funcionamiento del estándar ocupo únicamente unos días de practica con ejemplos de aplicaciones que cumplen con el estándar, suministrados por el tutor. Después de la comprensión de la estructura del estándar, que es básicamente la explicada en el apartado 3.2.2 se procedió inmediatamente a la adaptación de la aplicación (ver Apartado 5.4).

5.2 Diseño de la Aplicación

Aunque durante la redacción de la memoria se ha visto necesaria la separación en dos apartados distintos, la creación de la aplicación, el trabajo resumido en este apartado se realizo simultáneamente al del apartado siguiente (5.3). Por ello, en ocasiones se indican referencias entre ambos.

Afortunadamente, las herramientas de C# ofrecen todas las facilidades para la creación de aplicaciones similares a las planteadas en los bocetos ya expuestos en el apartado 4.2.2. Aunque con ciertos cambios visuales, las funcionalidades implementaron en la aplicación siguiendo los diagramas de flujo, también expuestos en el apartado 4.2.2, con el añadido de otra funcionalidad. Este añadido tiene un fin solamente ilustrativo, con el se pueden apreciar las partes en las que se ha dividido el código del algoritmo. En el apartado 5.3 se explican detalladamente estas partes (ROI, Enhancement, Vein Pattern, Skeletonization, Lines Pattern y Circles Pattern).

Para comprender correctamente la estructura de la aplicación visual creada es necesario conocer el flujograma completo de su funcionamiento; ver Figura 23.

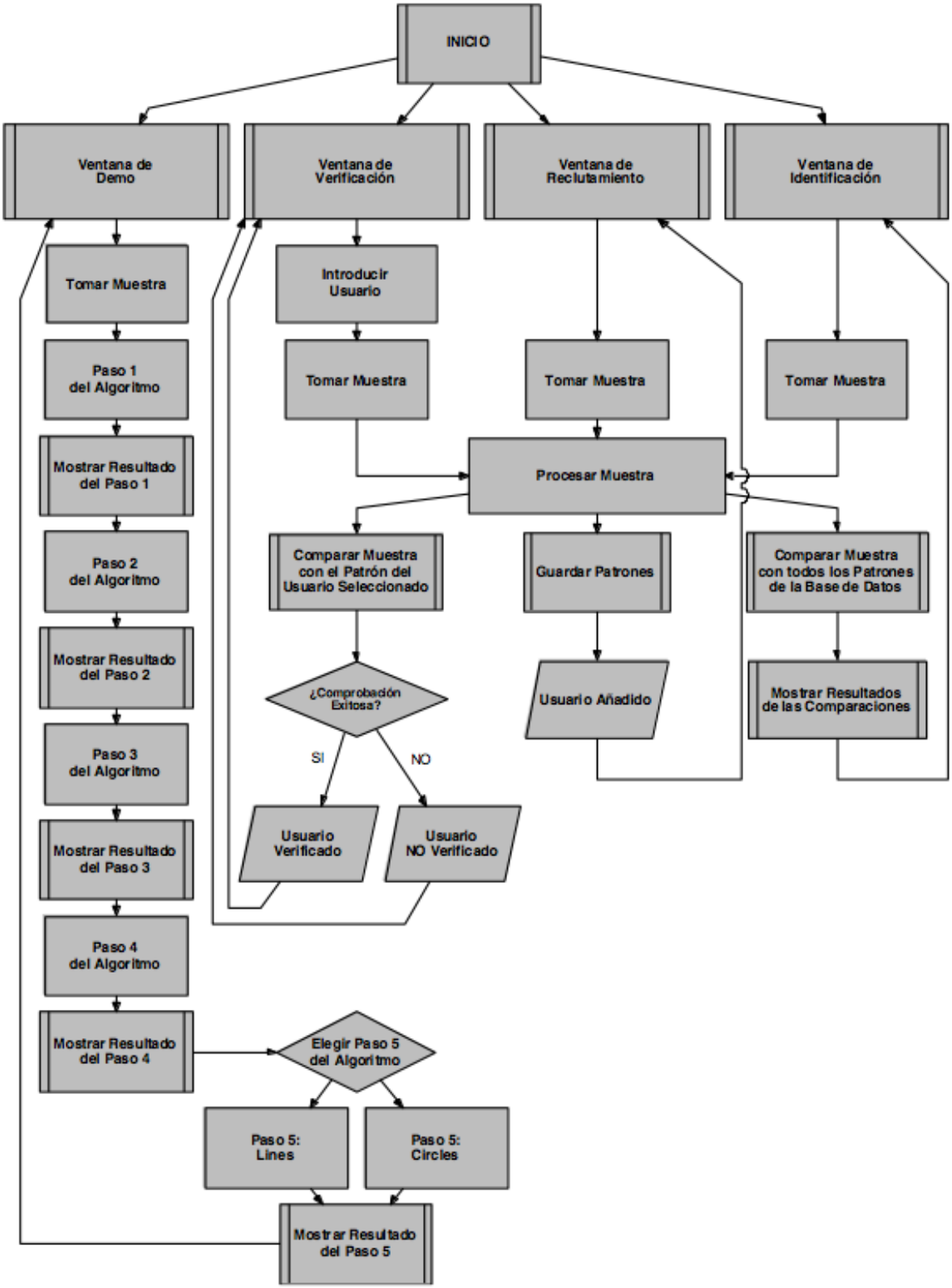

Figura 23. Flujograma de la Aplicación completa

La aplicación se divide en 4 ventanas separadas, implementadas por medio del control *Tab Control* (ver selección azul en Figura 24). Un usuario de la aplicación puede acceder a cualquiera de las cuatro funcionalidades de la aplicación en cualquier momento. Es posible abandonar un proceso a medias, proceder con otro y después reanudar el primero desde donde se haya abandonado. A continuación se van a describir las cuatro ventanas.

5.2.1 Demo

Esta es la ventana de la funcionalidad puramente ilustrativa de la aplicación que se comentaba antes en este capítulo. En ella se aprecian los pasos internos del algoritmo y el orden en el que se aplican.

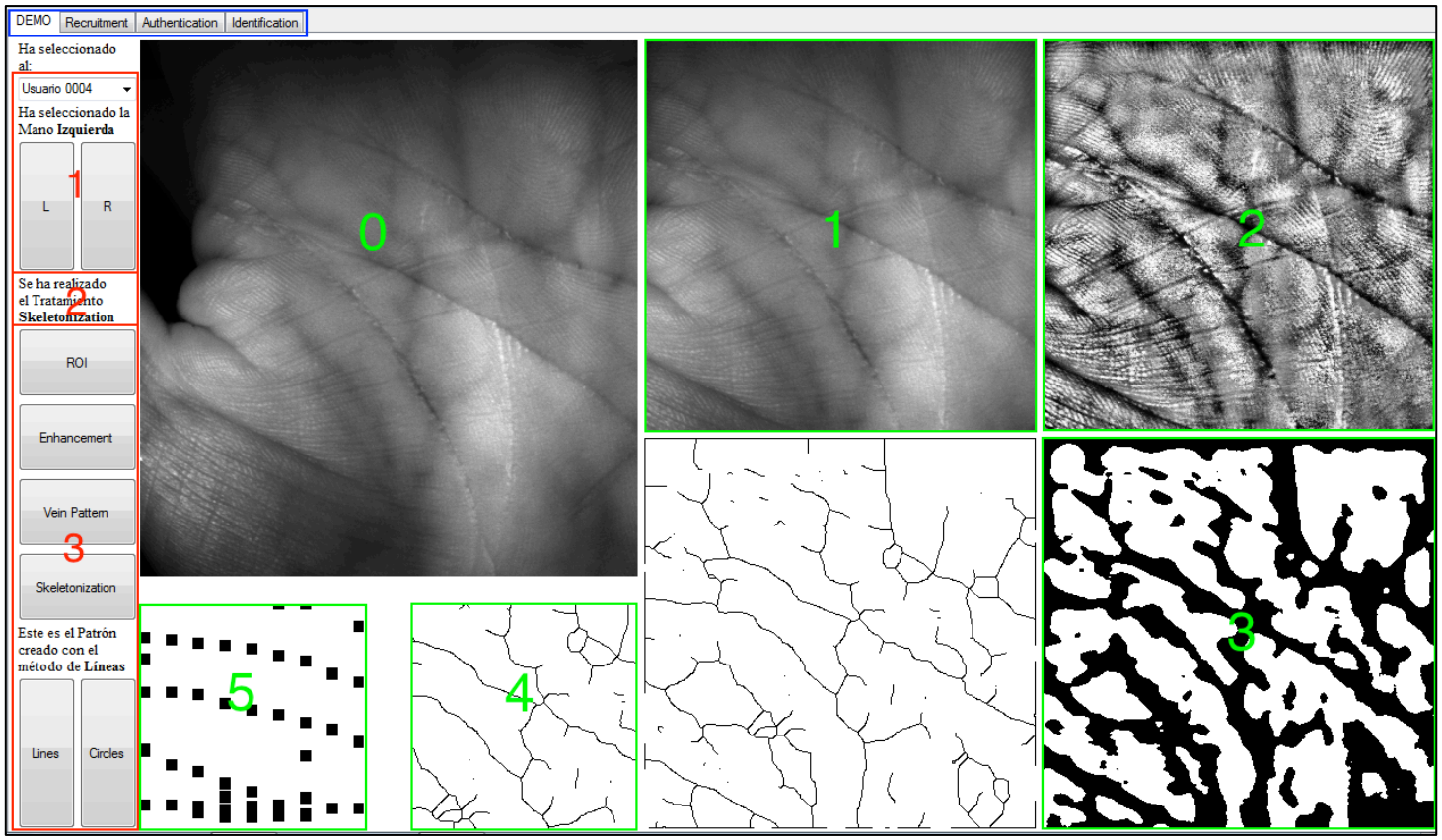

Figura 24. Aspecto de la funcionalidad Ilustrativa del algoritmo, Demo

Se pueden ver en la figura las distintas partes de la ventana Demo:

- Vienen marcadas en **rojo** las acciones que puede realizar el usuario. El recuadro número 1 corresponde a la parte de toma de muestra, ilustrada en el diagrama de flujo (Figura 23). Están seleccionados en este recuadro, el control desplegable de selección de usuario y los botones para la selección de la mano del usuario. El recuadro número 2 contiene la información que va guiando al usuario durante el proceso. Después de la selección del usuario, en la caja numero 0 aparece la muestra seleccionada. A

continuación se puede proceder al tratamiento de la muestra con los controles contenidos en el recuadro número 3. Estos controles corresponden a los pasos del 1 al 5 ilustrados en el diagrama de flujo.

- Los resultados de estos pasos o tratamientos sobre la imagen de la caja 0, están ilustrados en las cajas **verdes**. El resultado del Paso 1 del flujograma o ROI, se refleja en la caja número uno, el del Paso 2, en la caja 2 y así sucesivamente. En la caja 5 se representará el resultado del tratamiento *Lines* o *Circles* dependiendo de cuál sea seleccionado por el usuario. Todos los pasos se describen con detalle en el apartado 5.3.

5.2.2 Reclutamiento (Recruitment)

En esta ventana se realiza el proceso de reclutamiento de patrones de los usuarios. En este proceso los patrones son guardados en la base de datos para su posterior uso en las demás funcionalidades de la aplicación.

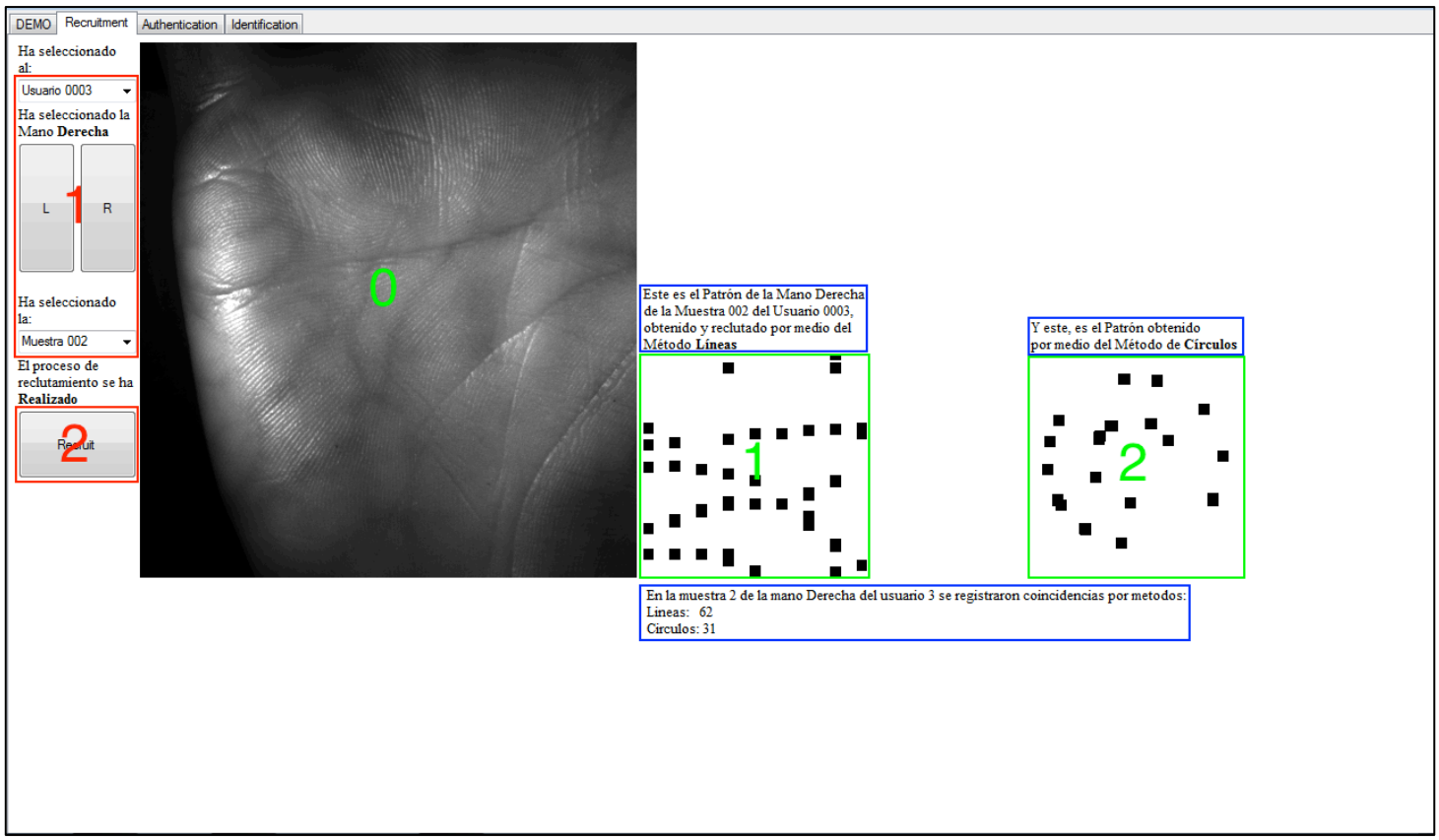

Figura 25. Aspecto de la funcionalidad de reclutamiento o Recruitment

Las secciones que forman parte del proceso de reclutamiento son:

- El recuadro **rojo** numero 1 corresponde, como en la funcionalidad de Demo, a la parte de toma de muestra del diagrama de flujo. En esta funcionalidad la toma de muestra

dispone de una acción adicional que es la selección de la muestra que se esta reclutando. En esta aplicación se puede reclutar hasta 4 muestras por cada mano del usuario, aunque este número es fácilmente ampliable. Cuanto mayor es el número de muestras con las que el sistema crea los patrones, más seguro es el sistema.

Como en la parte de Demo, tras la selección de la muestra está aparece en la caja número 0 y entonces, se puede proceder a aplicar a esta el algoritmo. Durante este proceso todos los tratamientos se realizan uno tras otro sin mostrar resultados intermedios al pulsar el botón del recuadro 2. Por ultimo, se guardan los patrones obtenidos en la base de datos.

- Los recuadros **verdes** muestran el resultado final de la aplicación del algoritmo a la muestra. El recuadro numero 1 muestra el resultado del algoritmo con el tratamiento final *Lines* y el recuadro 2, el tratamiento *Circles*.
- El texto, del interior de los recuadros azules, da información sobre que muestra se esta reclutando y algunos datos sobre sus patrones, como el numero de puntos de interés que se ha encontrado en cada uno (en al apartado 5.3 se explica cómo se consiguen los puntos de interés en los dos tratamientos).

La aplicación creada en este TFG tiene principalmente una finalidad didáctica y de investigación. Por ello, algunos de los aspectos que aparecen en esta lo hacen de una manera que no sería aceptable en una aplicación comercial. Por ejemplo, durante el proceso del reclutamiento, en una aplicación comercial, no se verían los patrones reclutados en un proceso ni tampoco datos como el número de puntos de interés. Estos datos se tendrían que ocultar para así evitar posibles intentos de fraude con esta información.

5.2.3 Verificación (Authentication)

En esta ventana los usuario pueden realizar el proceso de verificación. Una vez más, este no es un proceso de verificación que se vería en una aplicación comercial. En una aplicación comercial, durante la verificación, la muestra solo se enfrenta al patrón de ese usuario. Sin embargo, en la aplicación de este TFG es posible enfrentar cualquier muestra de la base de datos con cualquier otra y visualizar el resultado de esta simulación de verificación. Para tal fin, los procesos de introducir usuario y la toma de muestra, del diagrama de flujo, se han fusionado en esta funcionalidad.

Se explican, a continuación, las partes de la ventana de verificación:

- Como se puede ver en la Figura 26, no existe una parte de introducción de usuario. Se simula su funcionamiento con la serie de controles que aparece encerrada en el recuadro **rojo** número 1. De igual manera, en el recuadro rojo número 2 se simula la "selección automática" del patrón a partir del usuario "seleccionado" en el recuadro 1. El recuadro número 3 encierra el botón que pone en marcha la comparación de la muestra y el patrón que se hayan seleccionado.

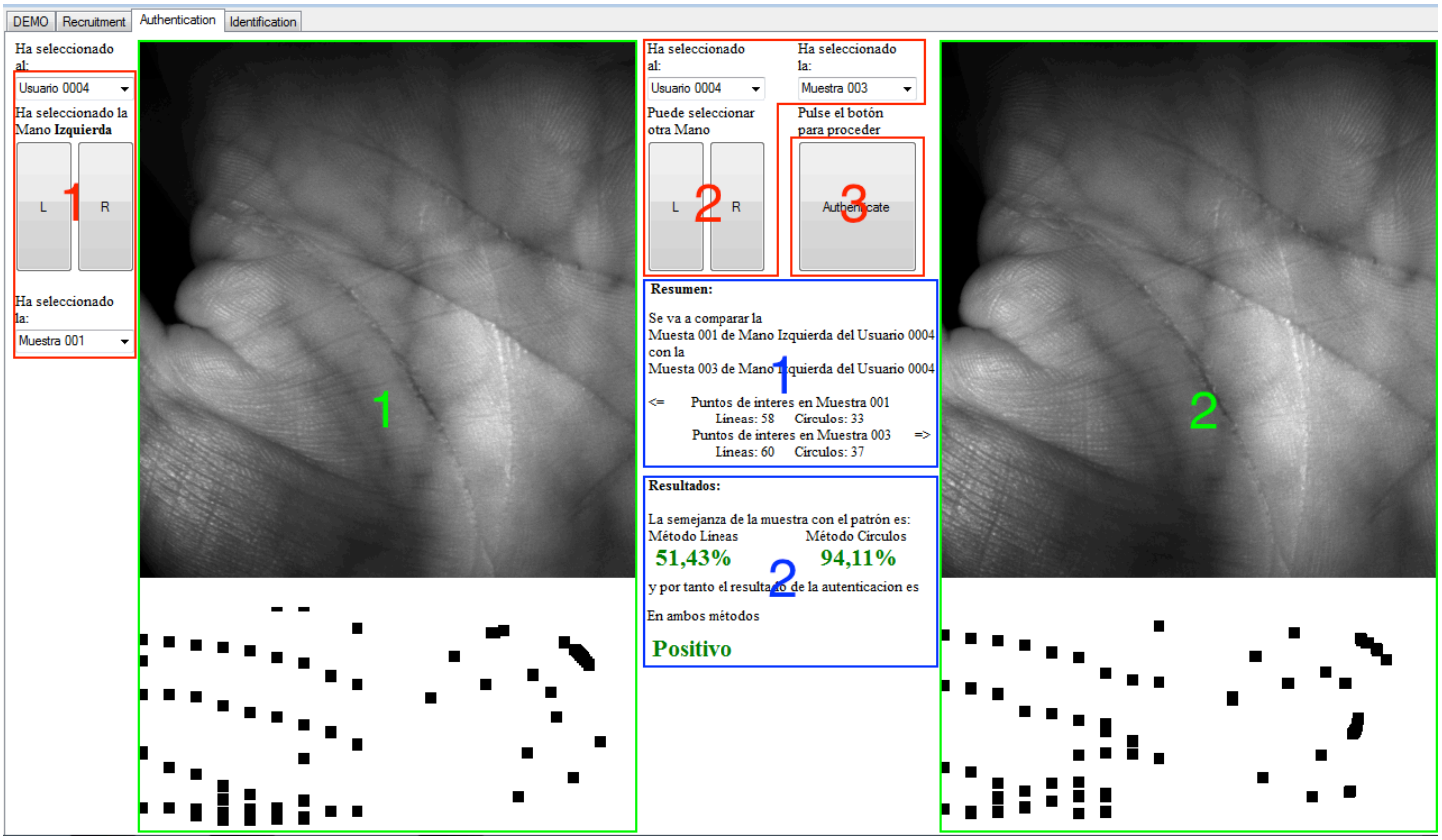

Figura 26. Aspecto de la funcionalidad de Verificación o Authentication

- En los recuadros **verdes**, 1 y 2, se muestran respectivamente la muestra y el patrón elegidos para la verificación . Se muestra en cada recuadro la imagen de la muestra (arriba), el patrón por el tratamiento de *Lines* de esa muestra (abajo, izquierda) y el patrón por el tratamiento de *Circles* (abajo, derecha).

- El texto del recuadro **azul** número 1 contiene información acerca de las dos muestras que se han seleccionado, como a que usuario pertenecen o cuantos puntos de interés contienen sus patrones.

  El recuadro número 2 muestra los resultado de la comparación, indica el porcentaje de similitud de la muestra con el patrón, y también el resultado positivo o negativo de esa comparación. La decisión de si el resultado es positivo o negativo depende del umbral seleccionado como se explica en el apartado 2.3.

Como ya se ha comentado en el apartado de reclutamiento, la aplicación muestra algunos aspectos que no serían aceptables en una aplicación comercial. En esta funcionalidad, una aplicación comercial no podría hacer visibles los patrones guardados en la base de datos ni tampoco ofrecer ninguna información respecto a la comprobación excepto la de la decisión tomada por el sistema, positivo o negativo.

5.2.4 Identificación (Identification)

La funcionalidad de esta ventana es la de identificar a un usuario de entre todos los reclutados en la base de datos a partir de una muestra.

Al igual que en la funcionalidad del apartado anterior, la introducción de una muestra se simula por medio de la selección de una imagen desde la base de datos. Después de la selección de la muestra, se procede a la comparación de esta con todas los patrones de la base de datos y a mostrar los resultados de esa comparación.

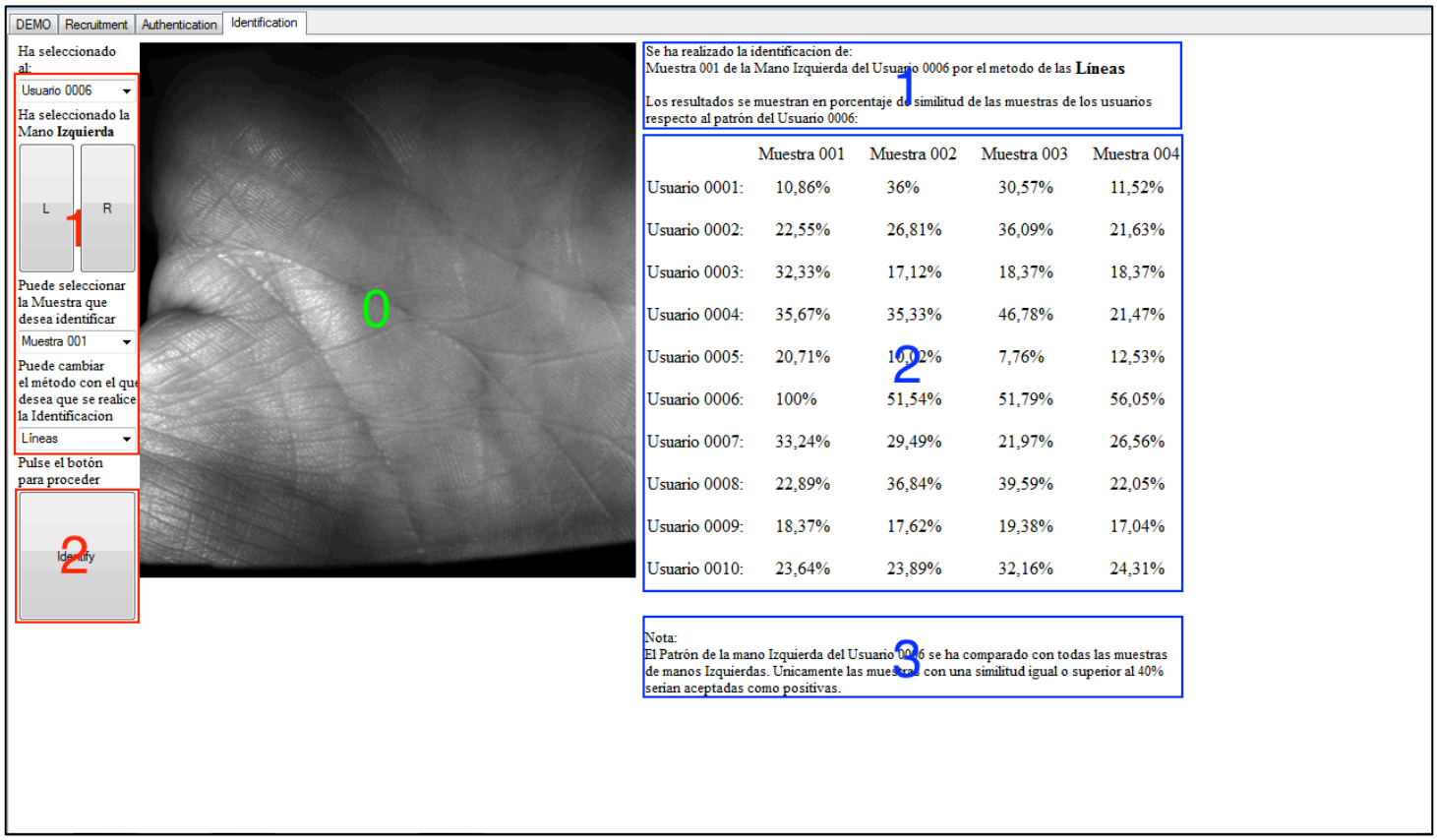

Figura 27. Aspecto de la funcionalidad de Identificación o Identication

Las partes en las que consiste la ventana de identificación son:

- El recuadro **rojo** número 1 encierra los controles de selección de muestra. Se puede observar que existe un control más que en las funcionalidades de reclutamiento y verificación. Como se explicará en el apartado siguiente esto es porque existen dos patrones distintos para cada muestra y por ello es necesario indicar con cual de ellos se desea realizar la identificación, lo resultados varían sensiblemente. Como en las demás funcionalidades, una ves seleccionada la muestra, esta aparece en la caja numero 0.

  El recuadro rojo número 2 encierra el botón que pone en marcha el proceso de comparación de la muestra seleccionada con el resto de muestras de la base de datos.

- Los recuadros **azules** muestran información referente a la identificación que se esta llevando a cabo. En el recuadro 1 se indica que muestra está siendo identificada y por medio de que método (*Lines* o *Circles*). En el 3 se explican las condiciones en las que una identificación se puede dar por correcta.

  En recuadro 2 contiene los resultados de las comparaciones que se han hecho con el método seleccionado.

En cumplimiento con las leyes de protección de datos comentadas en el apartado 1.2 se cuenta con el permiso expreso de todos los usuarios para el uso de sus parámetros biométricos en la investigación. Los nombres de los usuarios están totalmente desvinculados de estos paramentos guardándose en archivos diferentes.

5.3 Portado del Código, MATLAB a C#

Como se explica en el apartado 4.2.1, el método elegido para el portado del código es la transcripción manual de las funciones que componen el algoritmo. El proceso básicamente consiste en la lectura pormenorizada del código original y la búsqueda de funciones lo más similares posibles a las originales en el lenguaje de destino.

La aplicación creada en este TFG posee 2 grandes procesos, al igual que la aplicación original. El primero es el procesado de imágenes. En el diagrama de flujo de la Figura 23 está representado por la caja "Procesar Muestras" y los 5 pasos que lo componen conforman la funcionalidad Demo del diagrama. El segundo gran proceso de la aplicación es la comparación de imágenes que aparece en el diagrama como "Comparar muestra con el/los Patrón/es…". Estos son los procesos que debían ser portados de la aplicación original. Existen otros 3 procesos de apoyo ("Tomar Muestra", "Guardar Patrones" y "Mostrar Resultados") que también se comentarán en este apartado.

De las tres funcionalidades de la aplicación (Reclutamiento, Verificación e Identificación), todas utilizan el proceso de procesado de imágenes y solo las dos últimas utilizan el de comparación de imágenes. Por ello se decidió proceder en primer lugar con la el portado del proceso tratamiento de imágenes.

5.3.1 Procesado de Imágenes

Este proceso que se compone del algoritmo desarrollado en la Tesis Doctoral de José Enrique Suarez-Pascual: "Mecanismos de Captura y Procesad de Imágenes de Venas para la Identificación Personal"[24]. El algoritmo, creado por Dr. Suarez-Pascual, parte de fotos de las manos de usuarios tomadas con una cámara térmica fijada en una estructura especialmente diseñada para tomar fotos normalizadas. Las fotos son tratadas con varios filtros, a lo largo del proceso del algoritmo, eliminando partes que se consideran ruido hasta que únicamente queda representado el esqueleto de las venas de la mano. En el último paso

del algoritmo se aplican a este esqueleto distintas directrices, también diseñadas por Dr. Suarez-Pascual, que identifican puntos de interés en el esqueleto y se crean patrones únicos con estos puntos.

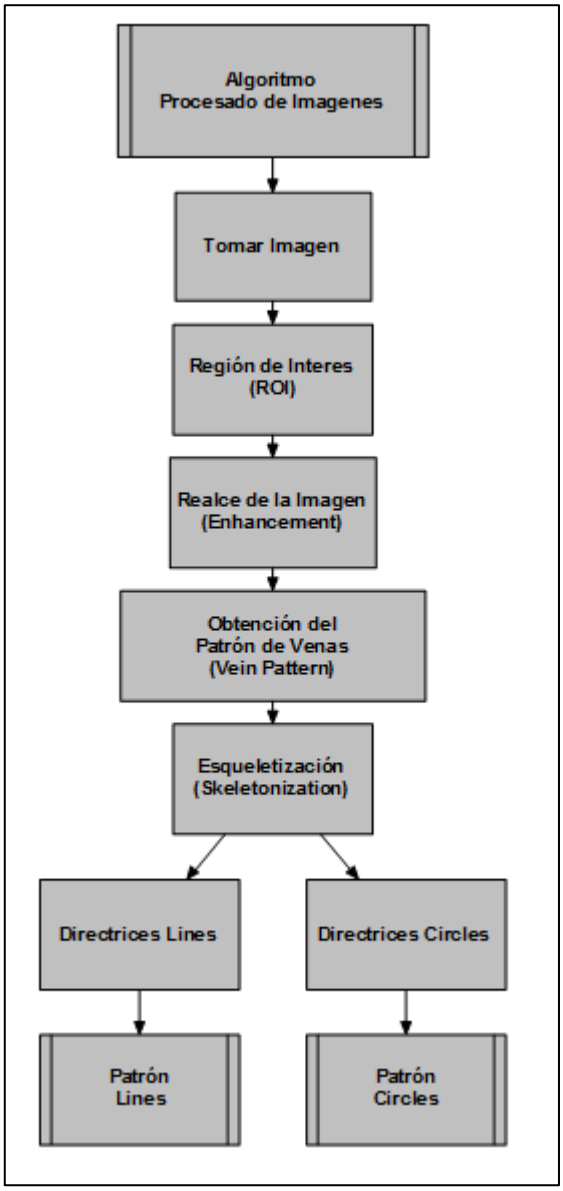

Figura 28. Diagrama de flujo del algoritmo creado por Dr. Suarez-Pascual

En el diagrama de la Figura 28 se observan los pasos de los que se compone el algoritmo. Excepto el paso de Tomar Imagen (la toma de imagen con cámara no es parte de este TFG), el resto de pasos están implementados en la aplicación y son los que se explican a continuación.

El proceso se compone de 5 pasos de los cuales el último, esencialmente son 2 pasos distintos aunque están en el mismo nivel jerárquico. Los tratamientos aplicados a las imágenes son los más parecidos que se han encontrado a los de la aplicación original y con los mínimos

cambios posibles a fin de que el resultado de la traducción del algoritmo sea únicamente una adaptación al nuevo lenguaje y no un algoritmo nuevo.

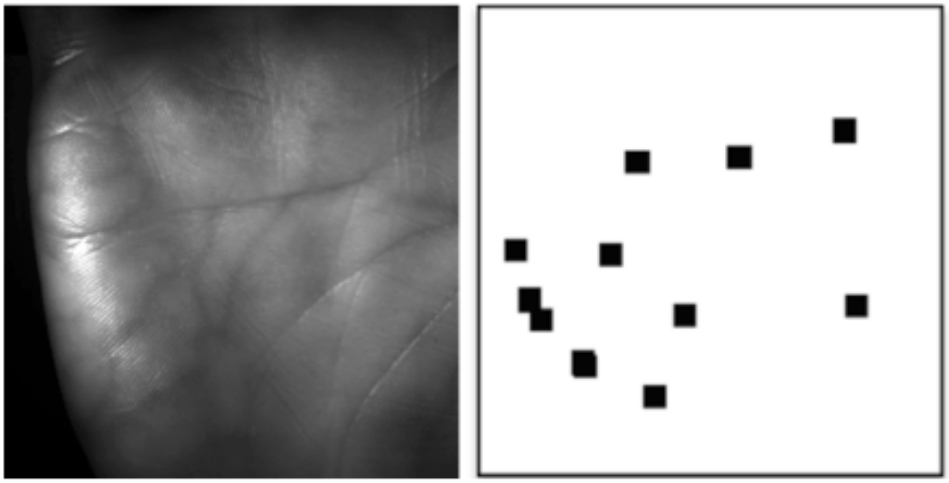

Figura 29. Muestra inicial y patrón final de esta muestra

5.3.1.1 Región de Interés (ROI)

El primer tratamiento que se aplica a la imagen original es el de la extracción del área de interés. El tamaño de la región de interés es de 350x350 pixeles, ubicados en el centro de la foto original (640x480 pixeles).

En el leguaje M, trabajar con imágenes significa trabajar con matrices. Las imágenes se cargan en matrices, cada celda de la matriz representa un pixel, y los tratamientos se aplican sobre estas. Por tanto en el lenguaje original este tratamiento consistía en la eliminación de celdas sobrantes y la conservación de las celdas de interés en una nueva matriz de 350x350.

En las bibliotecas de funciones instaladas para este TFG (Emgu CV) se encontró una función que realiza el mismo trabajo y es la función `ROI`.

```
mano.ROI = new Rectangle(64, 48, 350, 350);
```

Figura 30. Código de la función que realiza la extracción de la región de interés

La función realiza un corte en la imagen *mano* de 350x350 pixeles comenzando en el pixel (64,48) comenzando desde la esquina superior izquierda. Se muestra en la Figura 27 la imagen original de una muestra biométrica (izquierda) y la región de interés extraída de esta por medio de la técnica descrita.

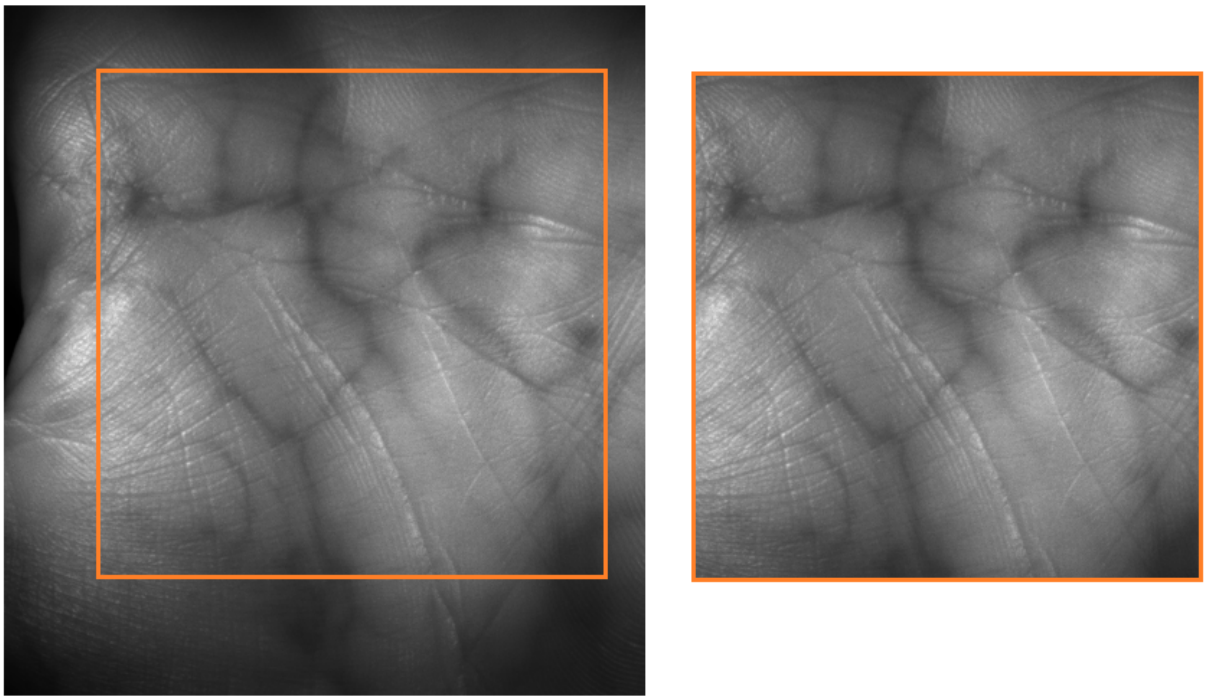

Figura 31. Resultado de la extracción de la región de interés

Puesto que no se trata de la aplicar ningún filtro a la imagen sino solo de recortarla, el resultado obtenido hasta este punto es el mismo que el de la aplicación original.

5.3.1.2 Realce de la Imagen (Enhancement)

El segundo tratamiento aplicado a la imagen (este se aplica a la imagen resultado del primer tratamiento) es un ajuste en el contraste de la imagen.

La función original, en lenguaje M, que realiza esta función es la *ADAPTHISTEQ* y lo hace ejecutando la técnica CLAHE (Constrast-Limited Adaptive Histogram Equalization). Esta técnica realiza ecualizaciones en varios histogramas que corresponden a las distintas partes en las que se divide la imagen sometida a la técnica. Esta es una versión más avanzada de la técnica AHE (Adaptive Histogram Equalization) que realiza una ecualización de histograma de toda la imagen.

La ventaja de realizar varias ecualizaciones en histogramas más pequeños es que de esta manera los realces son más significativos. Afectando cada ecualización un área más pequeña, intervienen menos datos en los cálculos y por tanto existe menos "ruido". La técnica redistribuye el brillo de la imagen mejorando notablemente su contraste.
Entre las bibliotecas de funciones Emgu CV, la función que realiza el tratamiento más **semejante** es la función `cvCLAHE`.

```
Emgu.CV.CvInvoke.cvCLAHE(imagen_IN, clipLimit, new System.Drawing.Size(15, 15), imagen_OUT);
```

Figura 32. Código de la función que realiza el ajuste de contraste

El lenguaje de origen funciona con un único tipo de datos, las matrices, pero no ocurre así con el lenguaje de destino (C#). Hasta este punto del proceso, Procesado de Imágenes, ya se han utilizado 2 tipos distintos. Las imágenes se cargan a la aplicación, desde la base de datos, con el tipo `Image<Gray, Byte>` (tipo propio de las bibliotecas Emgu) pero no todas las funciones pueden operar con este tipo. Por ejemplo, es necesario realizar un cambio de tipo en la imagen para poder someterla a la función descrita en este apartado. El tipo con el que opera esta función es `IntPtr`.

Es importante reincidir en que el tratamiento que se realiza sobre la imagen es **semejante** al que realiza la aplicación original. Aunque en teoría la función ejecuta la misma técnica que la original e incluso es posible ajustar los paramentos de las funciones a los mismos valores, el hecho de que los lenguajes operan con distintos tipos de datos hace que los resultados no sean idénticos.

En este punto del tratamiento la diferencia es prácticamente inapreciable ya que únicamente se ha aplicado un filtro a las imágenes pero al aplicar una serie de filtros, el resultado final es bastante diferente (ver Figuras 39 y 40)

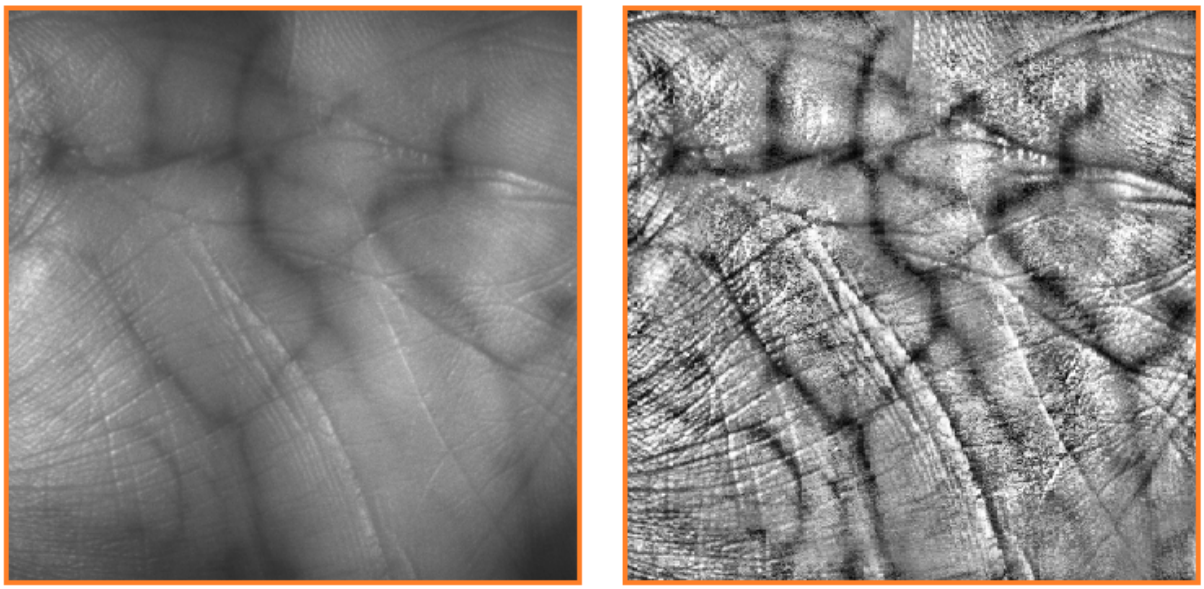

Figura 33. Resultado del paso Enhancement en la aplicación original

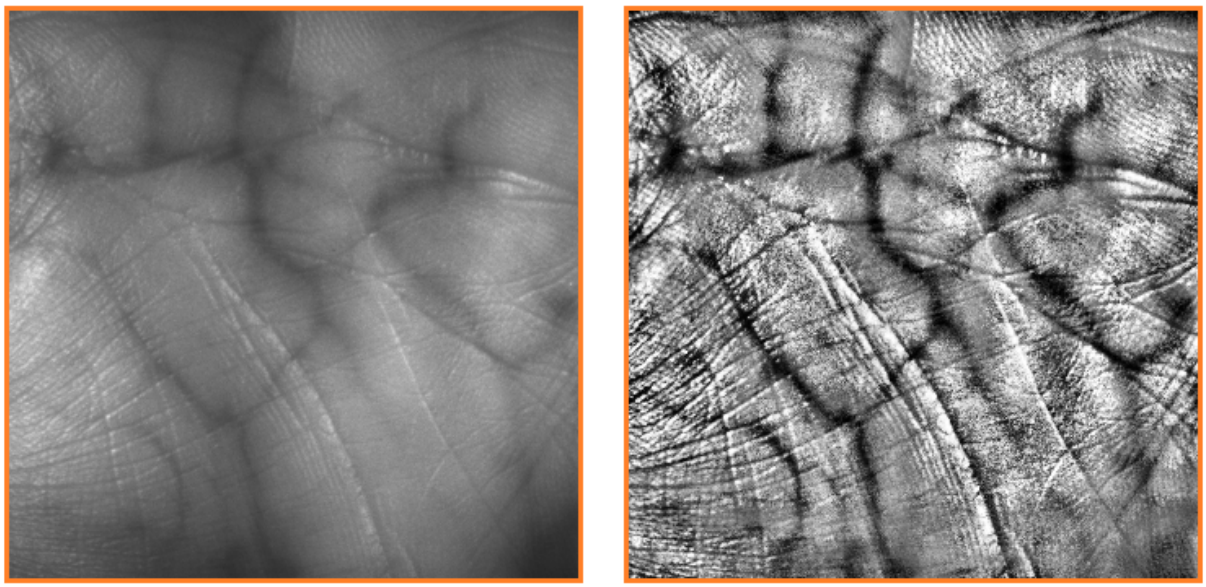

Figura 34. Resultado del paso Enhancement en la aplicación C#

En esta fase del portado del código se encontró un problema típico en el uso de software en fase *Beta*. El problema consistía que algunas de las librerías instaladas con el paquete Emgu.CV-2.4.9 Beta (x86) en realidad pertenecían a la versión de 64 bits. Afortunadamente las librerías afectadas no eran nuevas en esta versión. Por lo que, la solución consistió en la descarga de una versión más antigua pero estable de bibliotecas Emgu (Emgu.CV-2.4.2 (x86)), y la sustitución de las librerías afectadas por sus homónimas de la versión x86, que se encontraron en la versión más antigua de las bibliotecas.

5.3.1.3 Obtención del Patrón de Venas (Vein Pattern)

En este paso se aplican a la imagen, resultado del paso anterior, 3 filtros tras los cuales el resultado obtenido es el denominado patrón de venas. Los 3 filtros son considerados como un único tratamiento y por ello no se muestran resultados intermedios.

La serie original de filtros consiste en la aplicación de funciones *imextendedmax*, *imfilter* (gaussian), *medfilt2* propias del lenguaje M. Los filtros se aplican en este orden y producen los siguientes cambios en las imágenes:

- *Imextendedmax*. Realiza una binarización en la imagen. Hasta el momento las imágenes se representaban en escala de grises con hasta 256 niveles de gris. Esta función establece un umbral arbitrario en función del cual todos los grises pasan a considerarse o blanco o negro. Todos los grises que estén por encima de ese umbral pasan a ser negro y todos los que estén por debajo pasan a ser blanco.
  La función, de las bibliotecas Emgu, que realiza el cometido de la manera más parecida es `ThresholdBinary`.

```
patron._ThresholdBinary(new Gray(130), new Gray(255));
```

Figura 35. Código de la función que realiza la binarización de la imagen

- *Imfilter(gaussian)*. Función que aplica el filtro gaussiano a la imagen. El filtro o desenfoque gaussiano básicamente consiste en una ligera mezcla de los colores de los pixeles de una determinada vecindad.
  La función más semejante entre las bibliotecas Emgu es `SmoothGaussian`.

```
mano2 = mano.SmoothGaussian(3, 3, 1.5, 1.5);
```

Figura 36. Código de la función que aplica el filtro gaussiano a la imagen

- *Medfilt2*. Función que realiza una media con los valores de los pixeles de una determinada vecindad. La función de las bibliotecas Emgu encargada de aplicar este filtro es `SmoothMedian`.

```
patron = mano2.SmoothMedian(9);
```

Figura 37. Código de la función que aplica el filtro median

El problema surgido durante el portado de esta parte del código es que, la aplicación de los filtros descritos en el orden establecido en la aplicación original, produce resultados muy lejos de los esperados.

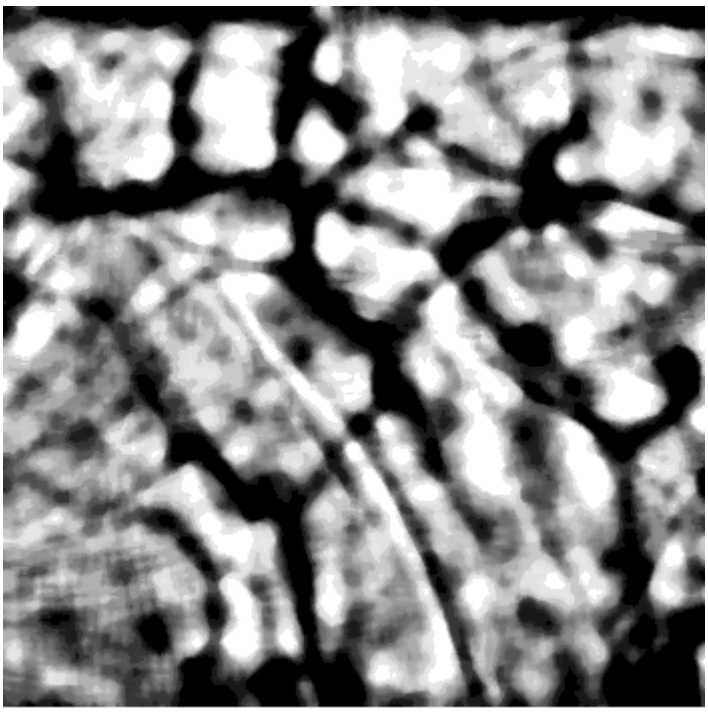

Figura 38. Resultado de la aplicación rigurosa del algoritmo hasta este punto

En busca de una solución para este problema se descubrió que alterando el orden de los filtros aplicados a las imágenes se produce un resultado más parecido al esperado basándonos en la aplicación original. Con el mismo objetivo se han tenido que modificar ligeramente los paramentos de las funciones explicadas antes. Se aumentó el umbral para la discriminación

entre grises (binarización) y se redujo el tamaño de las vecindades para la aplicación de los dos filtros de difusión (gaussiano, mediana).

Finalmente el orden de las funciones con el que los resultados son lo más parecidos a los originales es: 1ºSmoothGaussian (filtro gaussiano), 2ºSmoothMedian (filtro mediana), 3ºThresholdBinary (binarización).

En las figuras 39 y 40 se puede apreciar la diferencia entre los resultados obtenidos hasta este punto por las dos aplicaciones. Aunque se conserva la estructura básica del patrón obtenido, es evidente que tras la aplicación de 4 filtros, uno tras otro, en las dos aplicaciones, los resultados son bastante distintos.

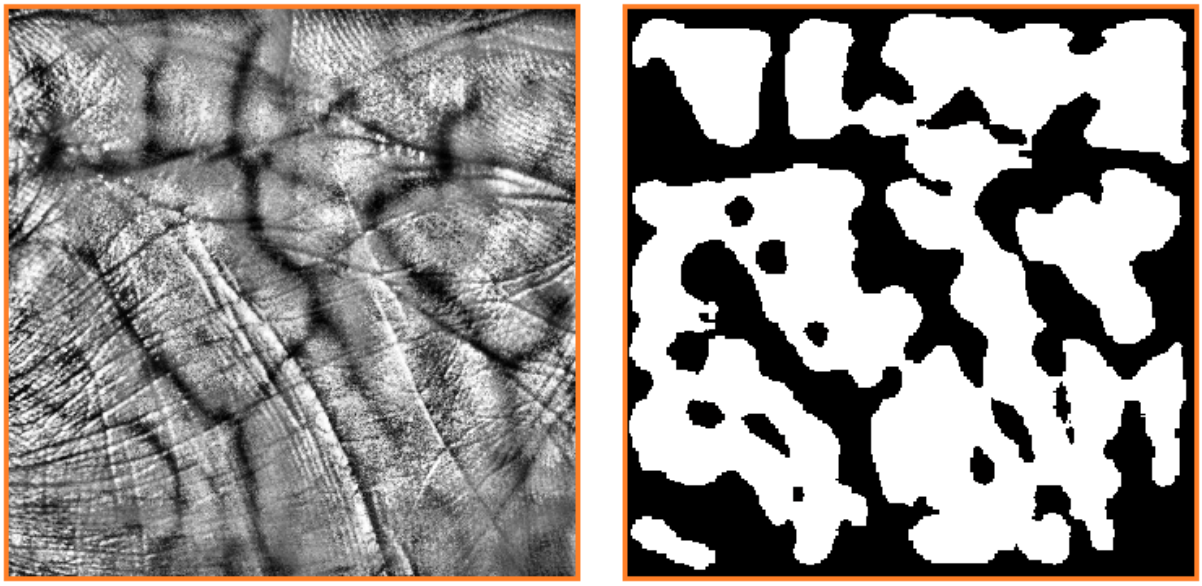

Figura 39. Resultado del paso Vein Pattern en la aplicación original

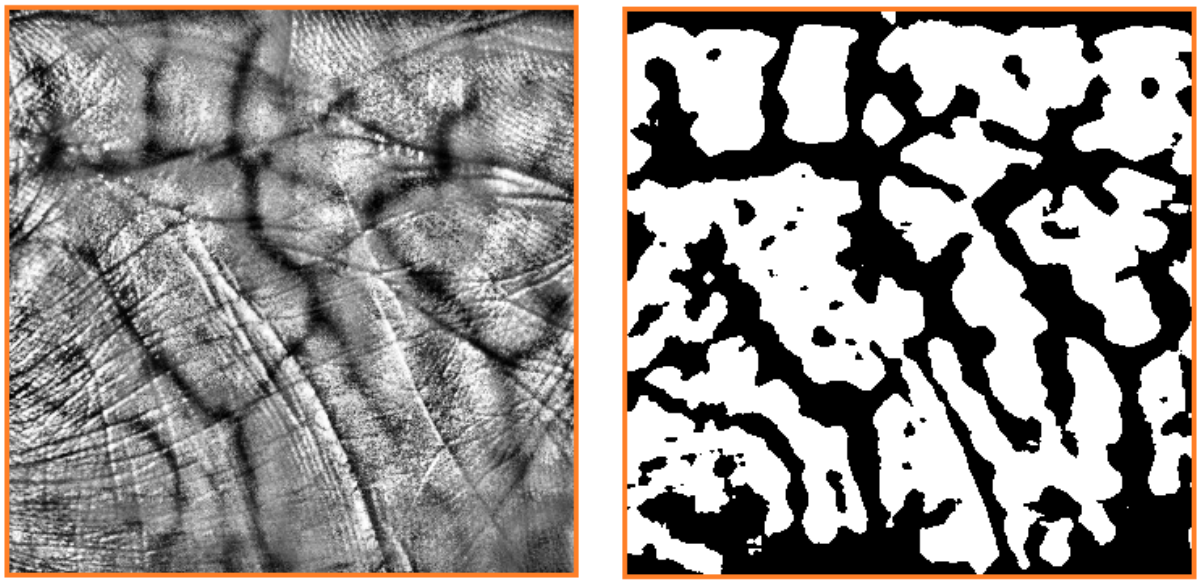

Figura 40. Resultado del paso Vein Pattern en la aplicación C#

5.3.1.4 Esqueletización del Patrón de Venas (Skeletonization)

En este paso, se afina la estructura del patrón de venas obtenida en el paso anterior y se vuelve a extraer una región de interés, aun más pequeña, del resultado de este tratamiento. A este proceso de afinamiento se le denomina esqueletización y en la aplicación original se realizaba por medio de la función *bwmorph*.

El problema surgido durante el portado de esta parte del algoritmo fue que, entre las bibliotecas de funciones utilizadas (Emgu) no se encontró ninguna función que realizara este proceso.

La única solución posible a este problema es la creación de una función para este cometido. Investigando las diferentes posibilidades por la red, se descubrió que lo que realmente hace la aplicación original es ejecutar el algoritmo Zhang-Suen sobre la imagen con lo que se decidió crear una función que ejecutase este algoritmo en C#.

El algoritmo Zhang-Suen[25] produce el efecto de esqueletización aplicado sobre imágenes binarias. En cada iteración va eliminando los pixeles más externos de la estructura sin permitir la separación de esta en partes más pequeñas.

Para la implementación de la función "propia" del algoritmo Zhang-Suen se decidió seguir la misma estructura que las bibliotecas Emgu. Como se explicó en el apartado 3.1.2, Emgu CV es únicamente un envoltorio .NET que permite el funcionamiento de funciones OpenCV (estas están escritas en C++) en C#. De igual manera la función que realiza el proceso del algoritmo Zhang-Suen fue escrita en C++, el código de las funciones se ha basado en el encontrado por la red durante la investigación sobre el algoritmo[26]. La función final ha sido empaquetada en una librería (un archivo .dll) e importada al proyecto de la aplicación. De esta manera es posible el uso de la función como si de otra función Emgu cualquiera se tratara.

```
Thinning.Thinning.thinning(IN, OUT, 1);
```

Figura 41. Código de la llamada a la función de afinamiento creada. Los argumentos son, IN: imagen que será sometida al tratamiento, OUT: imagen resultado del tratamiento, resto: argumentos auxiliares.

Después de la realización de la esqueletización de la imagen, se vuelve a extraer un área de interés. El tamaño del esqueleto de venas final es de 200x200 pixeles.

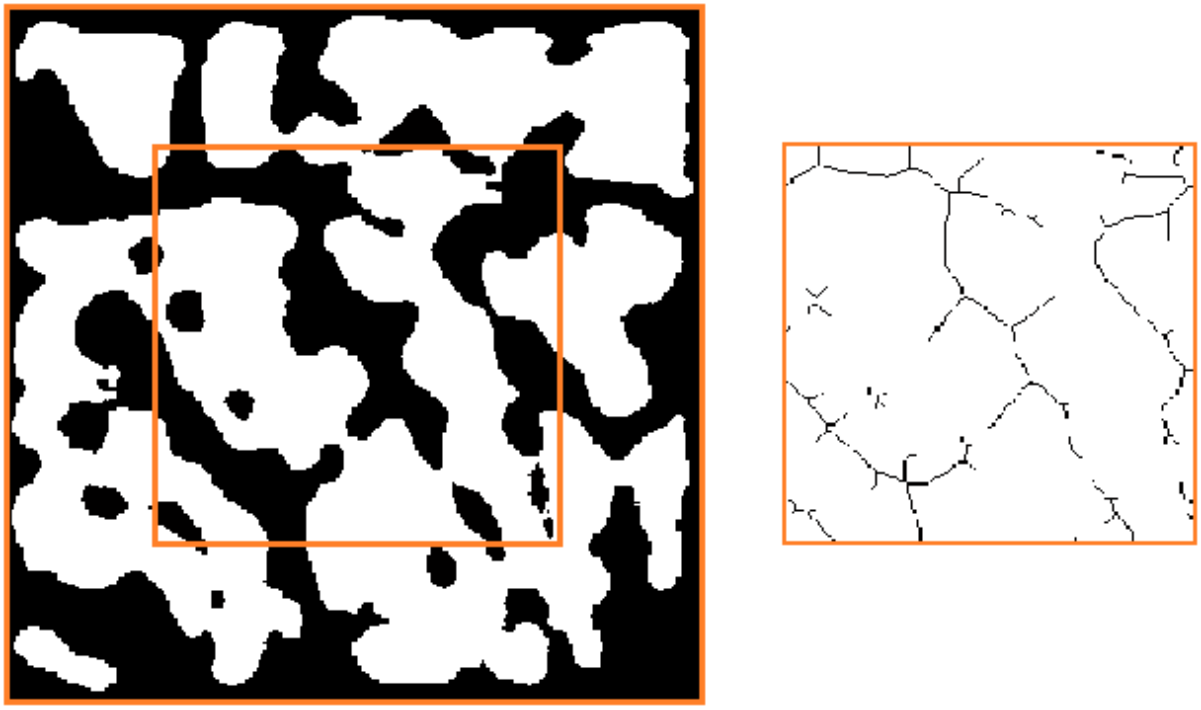

Figura 42. Resultado de la esqueletización en la aplicación original

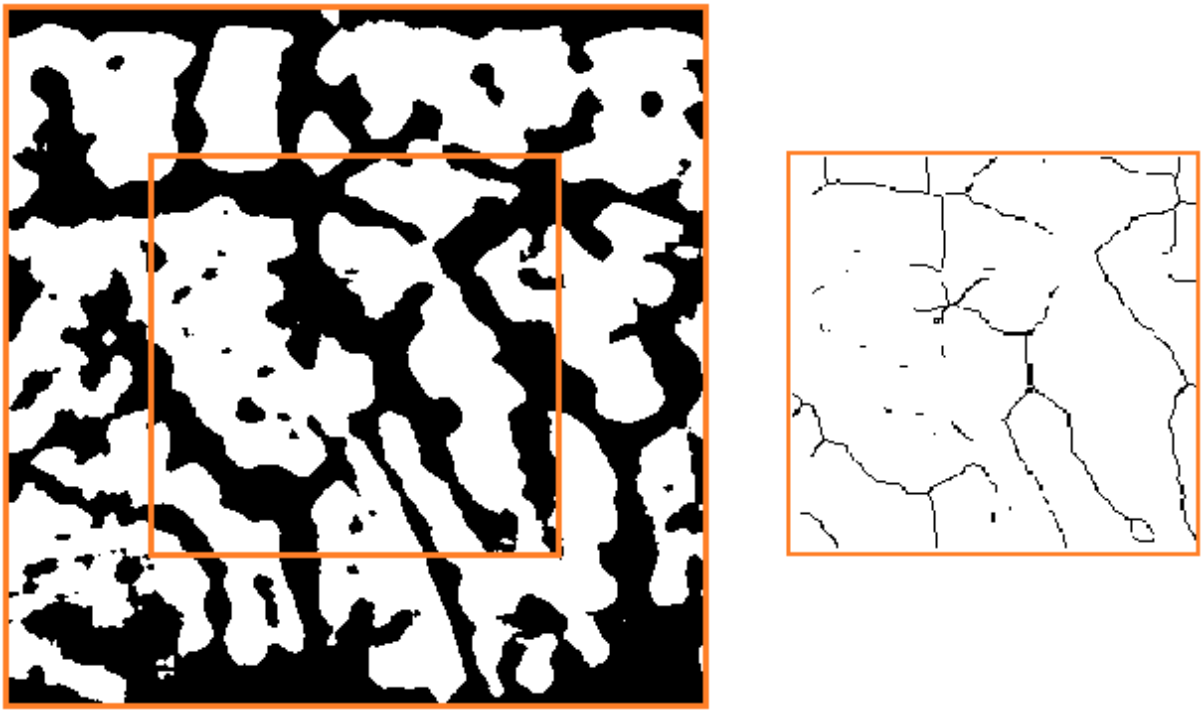

Figura 43. Resultado de la esqueletización en la aplicación C#

Una vez más se puede observar que aun compartiendo estructura básica, los resultados de las dos aplicaciones no son idénticos.

5.3.1.5 Tratamientos Lines y Circles

El último paso del proceso Procesado de Imágenes consiste en la creación de patrones definitivos. Estos se van a guardar para las comparaciones del proceso Comparación de Imágenes. Este paso no consiste en el uso de alguna función ya existente sino en la aplicación de ciertas directrices para la creación de los patrones a partir del esqueleto de venas. Los dos métodos para la creación de patrones parten del esqueleto de venas y aplican directrices distintas por que se crean dos patrones distintos a partir de cada muestra. Las directrices aplicadas en la aplicación original son:

- *Lines*. El esqueleto es superpuesto por 8 líneas horizontales (2 pixeles de grosor), equidistantes y dispuestas regularmente por el área del esqueleto.
- *Circles*. El esqueleto es superpuesto por dos círculos (2 pixeles de grosor), concéntricos, con diámetros 1/3 y 2/3 de longitud del lado de esqueleto.

En ambos métodos, los puntos en los que coinciden los pixeles que forman el esqueleto con los pixeles que forman las líneas (o los círculos) se consideran puntos de interés. Se crean nuevas imágenes que conservan los puntos de interés hallados y con los demás pixeles en blanco.
Las imágenes resultado de estos dos métodos tienen un tratamiento común. Al resultado de estas directrices se les aplica una función de dilatación (función *strel('square')*). Esta consiste en hacer más grandes los puntos de interés existentes en las imágenes. Concretamente tomando como centro estos puntos se crea alrededor de ellos cuadrados de un tamaño determinado (por ejemplo, 10x10 pixeles).

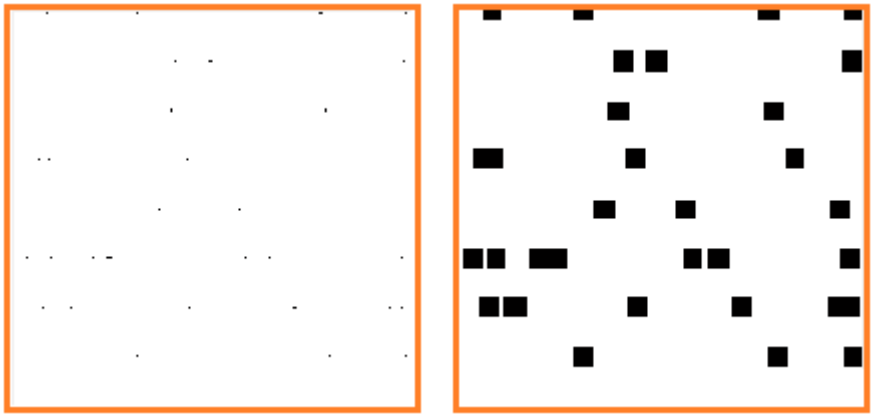

Figura 44. Resultado de la aplicación de las directrices del método *Lines* sobre el esqueleto (Izquierda) y resultado de la dilatación de los puntos de interés (Derecha)

En este punto es importante señalar que en la aplicación original el número de puntos de interés, creados con el método de *Lines*, siempre ronda entre los 30 y 45 (37 puntos en la Figura 44). El portado riguroso del algoritmo a la aplicación C# conlleva una reducción en el número de estos puntos hasta situarlo entre 20 y 30. Esto supone un problema ya que implica una menor disponibilidad de puntos para las comprobaciones en el proceso Comparación de Imágenes aumentando con ello los resultados de falso negativo y falso positivo.

En principio se pensó que es posible solucionar el problema aumentando la densidad de las líneas para así forzar más coincidencias con el esqueleto. Pero sorprendentemente aumentando el número de líneas prácticamente hasta el doble del original, el número de los puntos de interés creados apenas variaba.

La solución descubierta y finalmente implementada ha sido cambiar la dirección de las líneas sobrepuestas al esqueleto (pasan a ser verticales). A la vez se aumentó el número de líneas hasta 9. Originalmente las líneas se dibujaban, con espacios regulares de 25 pixeles, comenzando en el 1 y terminando en el 176 (1, 26, 51,…, 176, *201*) puesto que el 201 ya no existe (imágenes de 200x200 pixeles). Se ha reducido el espacio entre líneas a 24 pixeles permitiendo así incorporar una 9ª línea (1, 25, 49, 73,…, 193).

Con estas medidas el número de puntos de conseguidos en la aplicación C# se coloca a la altura del conseguido en la aplicación del lenguaje M.

Los resultados del método *Circles* no presentan grandes diferencias respecto a los de la aplicación original.

Para el portado de este paso del algoritmo se adoptó la misma estrategia del paso anterior. Se crearon funciones que realizaran las comparaciones del esqueleto con las líneas y los círculos en C++ y se empaquetaron en una librería para poder utilizarlas con una simple llamada como cualquier función Emgu. De hecho las funciones han sido empaquetadas en la misma librería de la función de esqueletización (*Thinning*).

```
Thinning.Thinning.metodoLineas(IN, OUT, 1);
```

Figura 45. Código de la llamada a la función creada del método *Lines*. Los argumentos son, IN: imagen que será sometida al tratamiento, OUT: imagen resultado del tratamiento, resto: argumentos auxiliares.

La función Emgu, utilizada para la dilatación de los puntos de interés encontrados es `cvDilate`.

```
Emgu.CV.CvInvoke.cvDilate(coincidencias_inv, coincidencias, square, 1);
```

Figura 46. Código de la función que realiza la dilatación de los puntos de interés

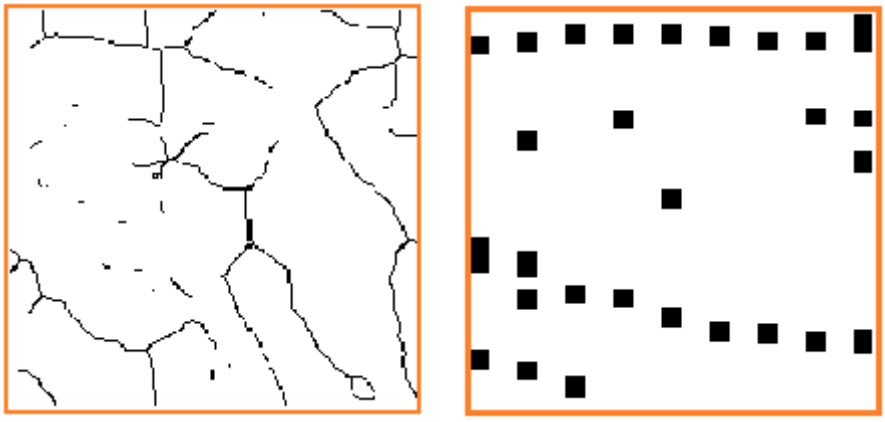

Figura 47. Patrón creado por el método *Lines*

Figura 48. Patrón creado por el método *Circles*

5.3.2 Comparación de Imágenes

Este proceso, usado por las funcionalidades Verificación e Identificación consiste en hallar la similitud de una imagen (muestra) con otra (patrón). Se aplica a los patrones conseguidos por el proceso de Procesado de Imágenes, tanto a los construidos con *Lines* como con *Circles*.

Existen numerosas funciones, entre las bibliotecas Emgu, que comparan imágenes pero ninguna que ofreciera los resultados que se necesitan para este proceso. Por ello y tratándose de una función bastante simple, se prefirió crear una propia. La implementación de esta se llevó con la misma estrategia que la del apartado anterior. Se escribió en C++ y se empaqueto con las demás funciones propias en la biblioteca *Thinning*.

```
porcentaje = Thinning.Thinning.comparador(Temporal1_Autenticacion, Temporal2_Autenticacion);
```

Figura 49. Código de la llamada a la función comparadora de imágenes

Los argumentos que necesita esta función son dos imágenes, la primera representa el patrón y la segunda una suma entre el patrón y la muestra.

En C# sumar dos imágenes binarias significa que las imágenes serán sobrepuestas y como resultado se tendrá otra imagen en la que únicamente aparecerán en negro los pixeles que compartan las dos imágenes superpuestas.

La función toma las imágenes Patrón y Suma y cuenta los pixeles en negro que hay en cada una. Conociendo esos números es fácil conseguir un porcentaje de semejanza, este es devuelto por la función. El porcentaje, resultado de estos cálculos, es la semejanza de la Muestra respecto al Patrón, es decir, cuántos pixeles de la Patrón también están en el Muestra (70,21% de semejanza de la Muestra respecto al Patrón en la Figura 49).

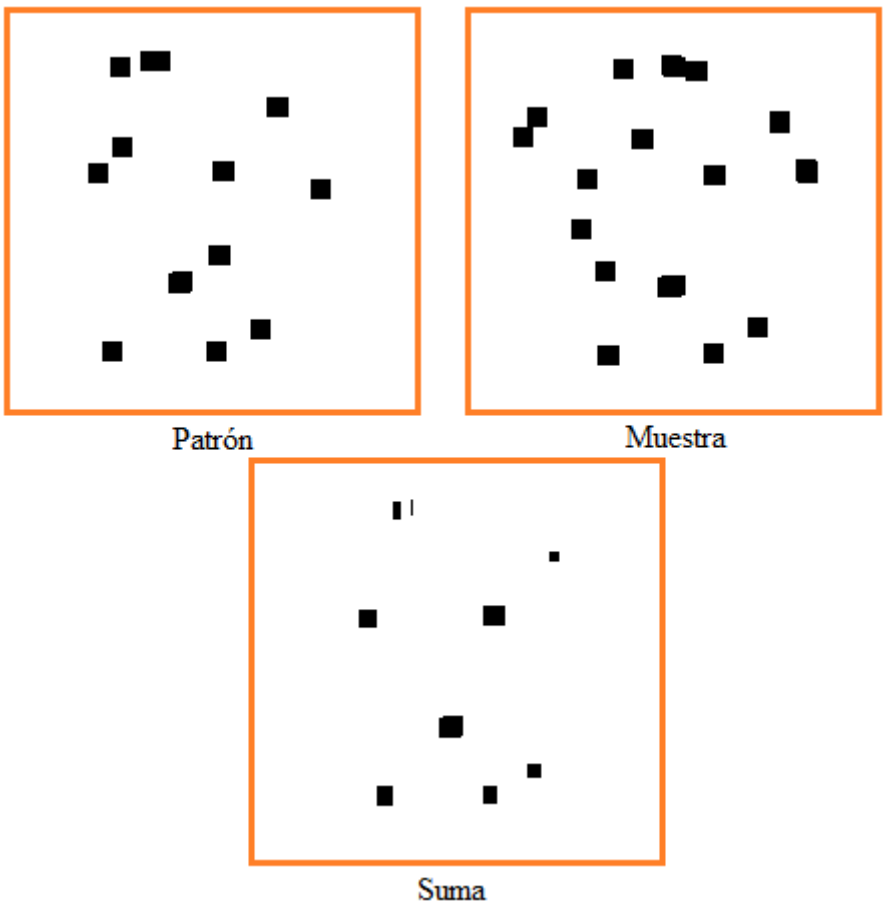

Figura 50. Resultado de la suma de dos imágenes binarias en C#

5.3.3 Procesos de Apoyo

Aparte de los 2 grandes procesos, la aplicación creada posee otros 3 que le sirven de apoyo. Se simula la funcionalidad de estos en la aplicación original aunque no se hallan portado como tal, C# posee métodos propios para ellos.

5.3.3.1 Tomar Muestra

En un sistema completo las muestras son introducidas a través de un sensor. Puesto que en este trabajo no se disponía de sensores, la introducción de una muestra se simula seleccionando una desde una "base de datos".

El proceso de selección de una muestra, desde la base de datos, consiste en la construcción del nombre de la muestra concreta requerida. Cada muestra posee un nombre que la identifica inequívocamente y es posible la construcción de este por medio de los controles representados en los recuadros rojos número 1 de las Figuras 24, 25, 26 y 27.

El nombre de las fotos sigue la siguiente estructura:

USUARIO_MANO_MUESTRA(_METODO*)

*La parte de método no está en todas las fotos, únicamente en el nombre de los patrones.

El ejemplo más completo es la construcción del nombre de un patrón que se realiza en la funcionalidad de Identificación, Figura 27. Si fuera necesario acceder, por ejemplo, al patrón del método *Circles* de la Muestra 3 de la Mano derecha del Usuario 7 se seguiría el siguiente proceso:

1. La selección del Usuario comienza la construcción del nombre de la foto:

   0007

2. La selección de la Mano añade este parámetro al nombre (_R):

   0007_R

3. La selección de la Muestra añade este paramento al nombre (_003):

   0007_R_003

4. La selección del Método añade este paramento al nombre (CIR):

   0007_R_003_CIR

La selección del último parámetro además añade al nombre la extensión del archivo (en este trabajo se ha usado .bmp para las fotos). El nombre completo es sumado a la ruta de la base de datos:

/…/ruta_base_de_datos/…/0007_R_003_CIR.bmp

Con esta información es posible la carga, a la aplicación, de la foto especificada a través de una función Emgu corriente.

```
Image<Bgr, Byte> patron = new Image<Bgr, Byte>("/.../ruta_base_de_datos/.../0007_R_003_CIR.bmp");
```

Figura 51. Función que carga las imágenes a la aplicación

5.3.3.2 Guardar Patrones

El guardado de patrones se lleva a cabo por una función elemental de Emgu. La única información que se necesita es el nombre bajo el cual se quiere guardar el patrón.

Para la creación del patrón se ha tenido que acceder anteriormente a la muestra a partir de la que se ha creado, por lo que ya se tiene la mayor parte del nombre, por ejemplo:

0007_R_003

A este nombre se la suma el parámetro especifico del método que se ha utilizado para crear el patrón (por ejemplo, *Lines*-LIN), la extensión (.bmp) y la ruta de la base de datos. Con este nombre se guarda el patrón:

```
imageBox2_2.Image.Save("/.../ruta_base_de_datos/.../0007_R_003_LIN.bmp");
```

Figura 52. Función que guarda las imágenes en la base de datos

5.3.3.3 Mostrar Resultados

Este proceso es el encargado de imprimir por pantalla la información obtenida en los procesos de Procesado y Comparación de Imágenes. En la información contenida en los textos informativos indicados en las Figuras 25, 26 y 27 únicamente varían los datos numéricos. Es fácil sustituir pequeños fragmentos en un *string* una vez que se tienen los datos numéricos adecuados. Este *string* se muestra por pantalla por medio del control *label*, propio de C#.

5.4 Adaptación de la aplicación al Estándar BioAPI

Acabado el proceso de portado de código y tras unos días de práctica con aplicaciones similares, con el estándar ya implementado, el alumno ya tenía soltura suficiente en el lenguaje C# como para proceder a la adaptación de la aplicación al estándar BioAPI.

El proceso de adaptación comienza tomando como plantilla el estándar, que ya existe en lenguaje C#, bajo la norma 30106-3 Object Oriented BioAPI. El trabajo consiste en rellenar algunas de las interfaces con el código del algoritmo de la aplicación y sustituir las llamadas a las funciones del algoritmo de la aplicación, por llamadas a las funciones propias del estándar. Con esto se consigue que un programador que esté revisando el código de la aplicación vea llamadas a las funciones, propias del estándar, *Process* o *Verify* pero no tenga forma de saber en qué consisten. Es decir, el código de las funciones explicadas en el capítulo 5.3 estará oculto para el programador, dentro de la estructura del estándar.

La estructura de las clases e interfaces del estándar tomado como plantilla es la siguiente:

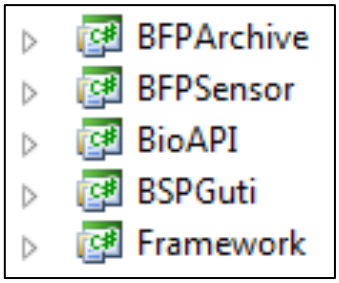

Figura 53. Plantilla del estándar

Estas clases definen distintas propiedades y estructuras que puede utilizar el estándar, no todas se usan en este trabajo.

- **BFPArchive**. Contiene las funciones necesarias para la manipulación de bases de datos.
- **BFPSensor**. Contiene funciones encargadas de la comunicación con los sensores conectados a la aplicación.

- **BioAPI**. Contiene las definiciones de algunas de las estructuras propias del estándar como el BIR (Estructura que encapsula datos biométricos para poder enviarlos entre las distintas clases del estándar) o el UUID (Identificador único estandarizado, identifica inequívocamente una muestra biométrica).
- **BSPGuti**. Contiene las funciones encargadas para el procesado y la comparación de datos biométricos.
- **Framework**. Interfaz que permite la comunicación de la aplicación con los componentes del estándar sin necesidad de conocer los procesos de sus funciones.

Entre todas las funciones existentes en estas clases, la aplicación creada requería la modificación de únicamente 4 de ellas. Todas las funciones restantes se dejan inalteradas en la estructura del estándar. Las funciones modificadas son:

- Función encargada de suministrar datos, de la base de datos a la aplicación y viceversa (`GetSingleBIR`), dentro de la clase BFPArchive. Siguiendo las recomendaciones del estándar, la función de la Figura 51, encargada de cargar en la aplicación las fotos de las muestras debe ser llamada desde aquí.

  Como se explica en el apartado 5.3.3.1, el proceso de toma de muestra se simula en este trabajo con una selección de la muestra desde una base de datos. Por ello, para poder seleccionar inequívocamente las muestras no es viable el cifrado de la base de datos ni la desvinculación de esta de la aplicación como recomendaría el estándar; los controles de selección de muestra deben permanecer en la aplicación.

  Sin embargo, en la adaptación del estándar se simula una desvinculación, de esta "base de datos", de la aplicación. El nombre de las muestras se sigue construyendo con los controles explicados en el apartado 5.3.3.1 pero antes de enviárselo directamente a la función `GetSingleBIR`, se cifra en un UUID. Este UUID (código aleatorio de 36 dígitos alfanuméricos) es encapsulado en un BIR y entonces enviado a la función `GetSingleBIR`. La función descripta el UUID, obteniendo el nombre construido de la muestra y entonces carga la muestra seleccionada. La muestra es convertida en *array* y enviada a la aplicación, encapsulada en otro BIR. Aquí la muestra es reconvertida a imagen y finalmente mostrada por pantalla.

  De esta forma la información enviada desde la aplicación a la función `GetSingleBIR`, y desde está a la aplicación esta siempre "cifrada". El envió de información, de esta manera, técnicamente cumple con el estándar.

- Función encargada de realizar el procesado de datos biométricos (`Process`), dentro de BSPGuti.

  La adaptación de esta función básicamente consiste en el traslado de las funciones que realizan la serie de tratamientos a las imágenes, explicados en el apartado 5.3.1 desde el código de la aplicación a esta función. En la aplicación, en lugar de llamar a estas

funciones se llama a la función `Process`, propia del estándar, y esta a su vez llama a las funciones encargadas del tratamiento de las imágenes.

De esta manera se consigue que desde la aplicación realmente no se sepa que tratamiento se le está aplicando a las muestras, que es lo que persigue el estándar.

- Las últimas dos funciones modificadas son las encargadas del proceso de comparación de muestras biométricas (`Verify` e `Identify`), dentro de BSPGuti.

  Ambas funciones, usadas respectivamente en las funcionalidades Verificación e Identificación, utilizan el proceso Comparación de Imágenes por lo que su adaptación es muy similar. Básicamente consiste en el traslado de la operación de suma de imágenes (5.3.2) y la función comparadora de imágenes (Figura 50) desde el código de la aplicación a esta función.

  Desde la aplicación se llaman las funciones `Verify` o `Identify` y estas a su vez realizan sus procesos llamando a las funciones del proceso Comparación de Imágenes haciendo imposible saber, desde la aplicación, qué procesos se siguen para la verificación de usuarios.

Como se viene insistiendo, la modificación de estas funciones con tal de adaptarlas al estándar consigue hacer "ciega" a la aplicación, frente a tratamientos y procesos a los que son sometidos los datos biométricos.

6 Pruebas

En el presente capitulo se van a comentar las pruebas para comprobar el correcto funcionamiento de las aplicación. El capítulo se divide en dos apartados en los que se resumen las pruebas durante la traducción del algoritmo y las pruebas en la aplicación final.

6.1 Pruebas durante el Desarrollo

Esta serie de pruebas se llevaron a cabo durante el proceso de adaptación del código original al nuevo lenguaje. A medida que se iba traduciendo el código y recreándose el algoritmo se iban haciendo pruebas.

Comparando datos intermedios de los algoritmos de MATLAB y C# no se obtenían resultados claros ya que siempre existe entre ellos diferencias por el cambio de plataforma (cambio de tipo de datos, implementación de funciones nativas en cada lenguaje, etc). Por ello, para comprar los resultados de distintas partes del algoritmo, inicialmente se realizaron comparaciones visuales. Fue de esta forma como se descubrieron las diferencias entre los resultados del paso *Vein Pattern* de la aplicación original y la adaptada.

La adaptación rigurosa del algoritmo provoca resultados muy distintos a los obtenidos en la aplicación original.

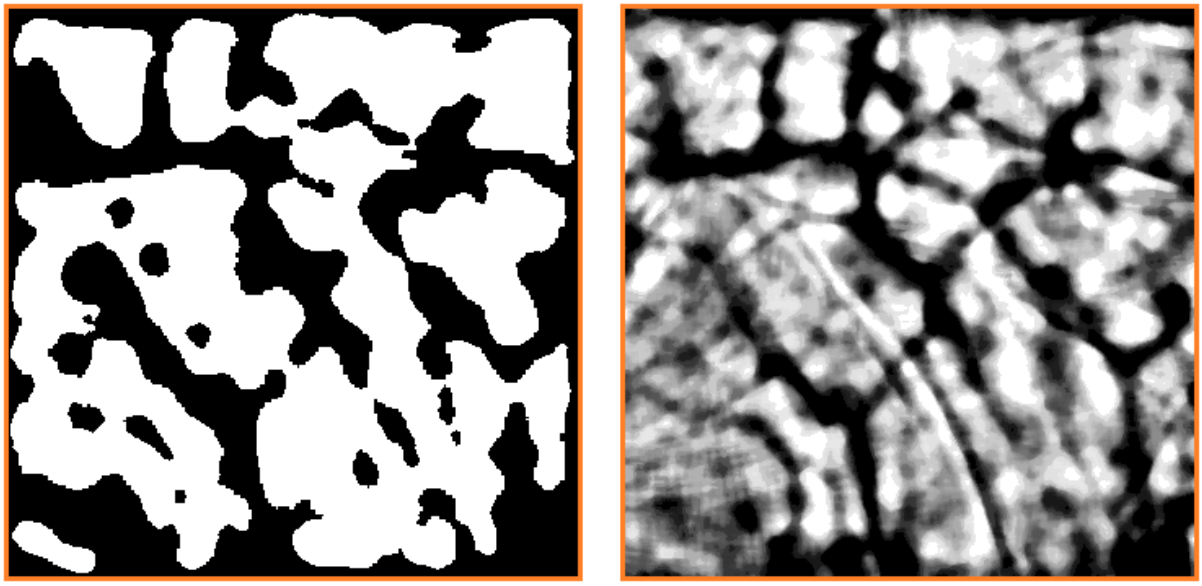

Figura 54. Detección de fallo en la adaptación del algoritmo (*Vein Pattern*). Resultado de aplicación original a la izquierda y resultado de aplicación C# a la derecha

Una vez detectado el fallo en los resultados de la aplicación obtenido se procedía al ajuste de los parámetros de las funciones y, como en este caso, a la alteración del orden de los tratamientos aplicados a las imágenes (5.3.1.3), hasta obtener un resultado lo más parecido al original posible.

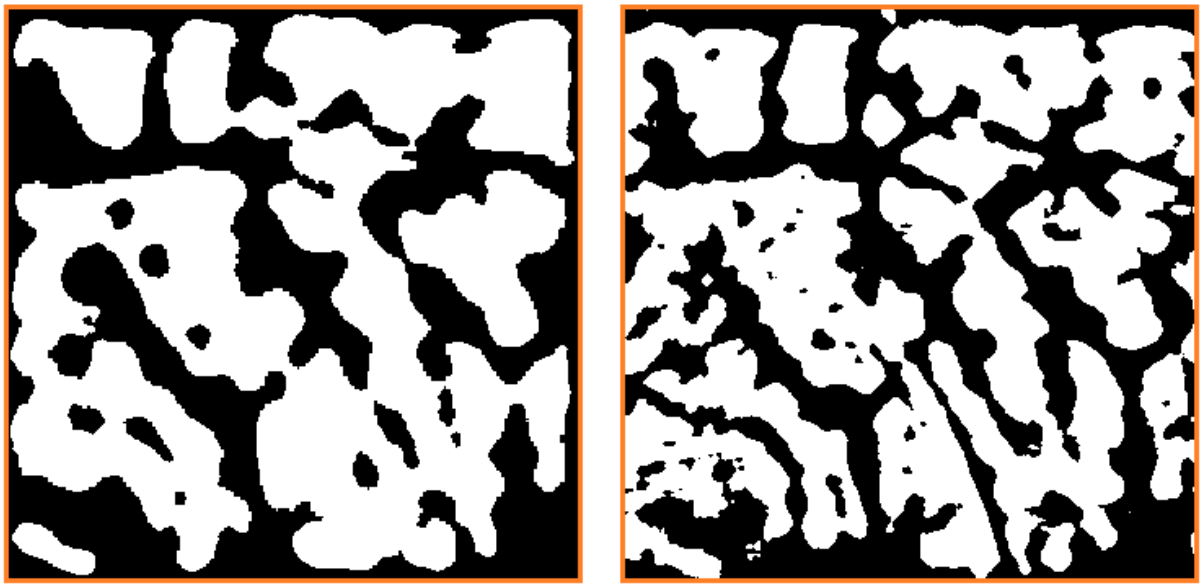

Figura 55. Resultado tras el ajuste de las funciones

Como se observa en la figura y ya se ha explicado en el apartado de la adaptación del código, los resultados no son idénticos a los de la aplicación original. El uso de distintos tipos de datos por parte de las dos aplicaciones, unido a los cambios existentes respecto al algoritmo original, conduce a inevitables diferencias en los resultados de las dos aplicaciones.

Teniendo esto en cuenta es lógico que se hayan encontrado diferencias en los resultados de otros tratamientos a los que se someten las imágenes. La solución, aparte del caso arriba comentado (*Vein Pattern*), siempre ha consistido en cambios mínimos, ajustes en los parámetros de las funciones responsables de los filtros, conservando con ello la estructura del algoritmo.

Como se ha comentado, durante esta fase del trabajo las comparaciones siempre han sido visuales, por lo que los cambios solo se introducían si se apreciaba alguna mejoría en 5 o más muestras. Ejemplos de estos cambios han sido, reducción de las vecindades para la ecualización de histogramas (parámetro del tratamiento *Enhancement*) o la implementación de nuevas líneas en el método *Lines* (cambio ya explicado en 5.3.1.5)

Traduciendo el algoritmo de forma separada a las funcionalidades de la aplicación se consiguió reducir considerablemente el tiempo de pruebas. Una vez completo el proceso Procesado de Imágenes (algoritmo), este se ha incrustado en todas las funcionalidades ya con la seguridad de un correcto funcionamiento.

6.2 Resultados en la Aplicación Final

Lo que se considera en este capítulo como Aplicación Final es el resultado de la adaptación al estándar de la aplicación perfectamente funcional, que ha superado las pruebas descritas en el apartado anterior.

Con estas pruebas superadas pasamos a analizar las pruebas referentes al funcionamiento de la aplicación con el estándar, el rendimiento del porte del algoritmo (5.4) y la calidad de la interfaz de usuario de la aplicación.

6.2.1 Estándar

Como se ha explicado en el apartado 5.4, la adaptación de la aplicación al estándar BioAPI no provoca cambios en cuanto al uso de esta. Todos los cambios son internos al código y por tanto un usuario de la aplicación no puede saber si esta lleva el estándar incorporado. En esta idea se basó la estrategia de pruebas de este apartado, es decir, se trataba de comprobar que el funcionamiento de la aplicación, con el estándar, fuese idéntico al de la aplicación sin él, que ya existía.

La introducción del estándar en la aplicación supone la introducción de una capa extra de llamadas, entre las funciones de tratamiento y comparación de imagen y los eventos de la aplicación. Las funciones en lugar de llamarse desde los eventos (así se hace en la aplicación sin estándar) se llaman desde la estructura del estándar.

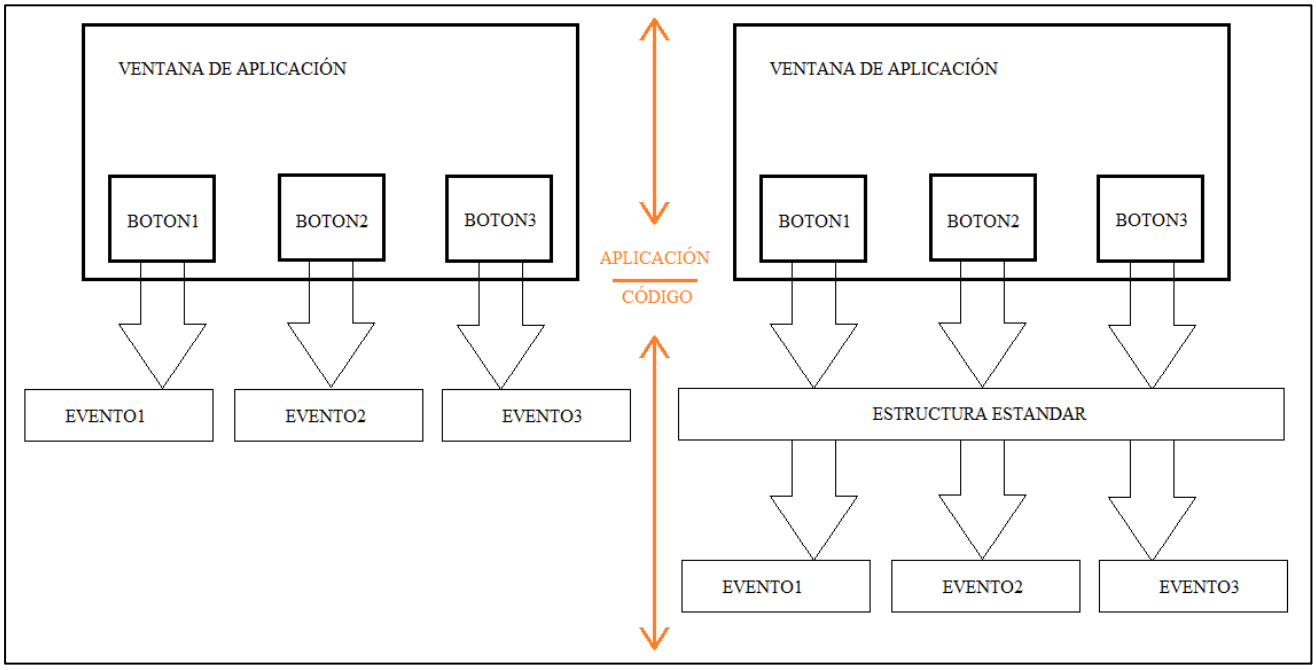

Figura 56. Estructura de la aplicación antes (izquierda) y después (derecha) de la implementación del estándar

La caja "Estructura Estándar" de la Figura 56 incluye toda la estructura explicada en 5.4 (llamadas propias del estándar, framework, BSPs, etc.).

Con esta capa extra, si la implementación está bien hecha (hipótesis de partida), todos los procesos de la aplicación deben funcionar igual que en la aplicación sin estándar por lo que se comprueban no a uno en las funcionalidades de la aplicación final.

6.2.1.1 Reclutamiento

En esta funcionalidad de la aplicación se comprueba el funcionamiento de los procesos Tomar Muestra y Procesado de Imágenes.

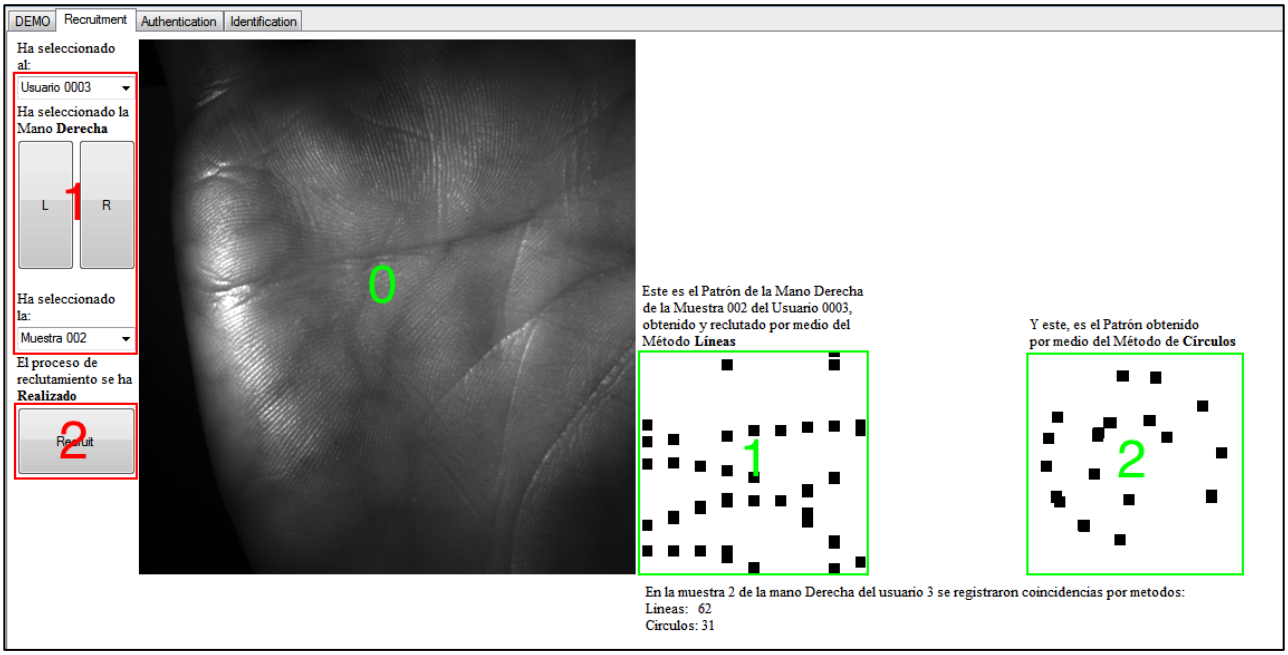

Figura 56. Reclutamiento en la Aplicación Final

Siguiendo las explicaciones de 5.2.2 sobre el funcionamiento de la ventana Reclutamiento, se selecciona la muestra y el hecho de que esta se muestre en la ventana 0 indica que el proceso de Toma de Muestra sigue funcionando tras la aplicación del estándar. Siguiendo con la funcionalidad de esta ventana, al pulsar el botón 2 (rojo) se realiza el proceso de Procesado de Imágenes. El hecho de que los patrones se muestren en los recuadros 1 y 2 (verde) y sean además idénticos a los obtenidos en la aplicación sin estándar (Figura 25) indica que este proceso funciona correctamente con el estándar.

6.2.1.2 Verificación

En esta funcionalidad de la aplicación se comprueba el funcionamiento de los procesos Guardar Patrones y Comparación de Imágenes.

Siguiendo las explicaciones de 5.2.3 sobre el funcionamiento de la ventana Verificación, se selecciona la muestra y en esta ocasión al mostrarse en las ventanas 1 y 2 (verde), además de comprobarse la Toma de Muestra se comprueba el proceso Guardar Patrones que ha tenido

lugar en la funcionalidad de Reclutamiento. De no haberse realizado correctamente el Guardar Patrones, en la funcionalidad Verificación no podrían aparecer los patrones que aparecen debajo de las ventanas 1 y 2 (verde).

Por otro lado se comprueba que el proceso Comparación de Imágenes (botón 2, rojo) funciona correctamente ya que los resultados obtenidos son los mismos que en la aplicación sin estándar.

En la versión final de la aplicación, se han eliminado los porcentajes de similitud de la ventana Verificación puesto que no es una característica usual en esta funcionalidad.

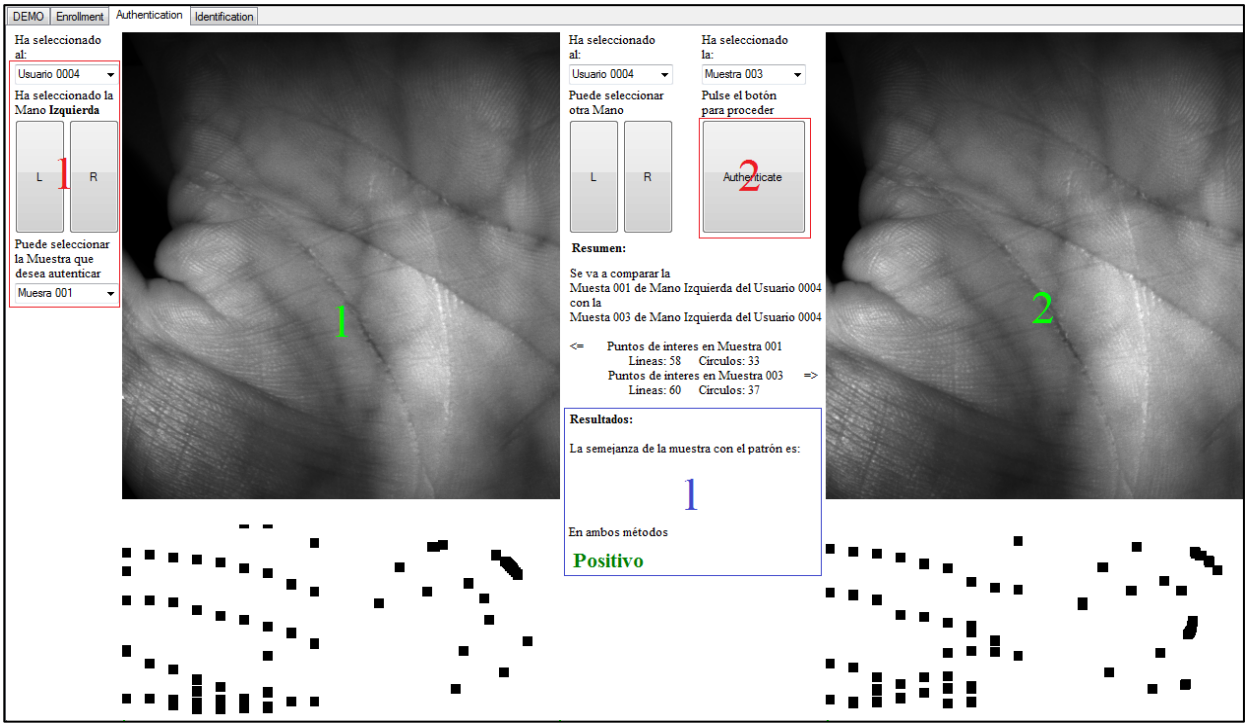

Figura 57. Verificación en la Aplicación Final

6.2.1.3 Identificación

En esta funcionalidad de la aplicación se vuelven a comprobar los mismos procesos del apartado anterior como la Comparación además del proceso Mostrar Resultados, que cobra especial importancia en esta funcionalidad.

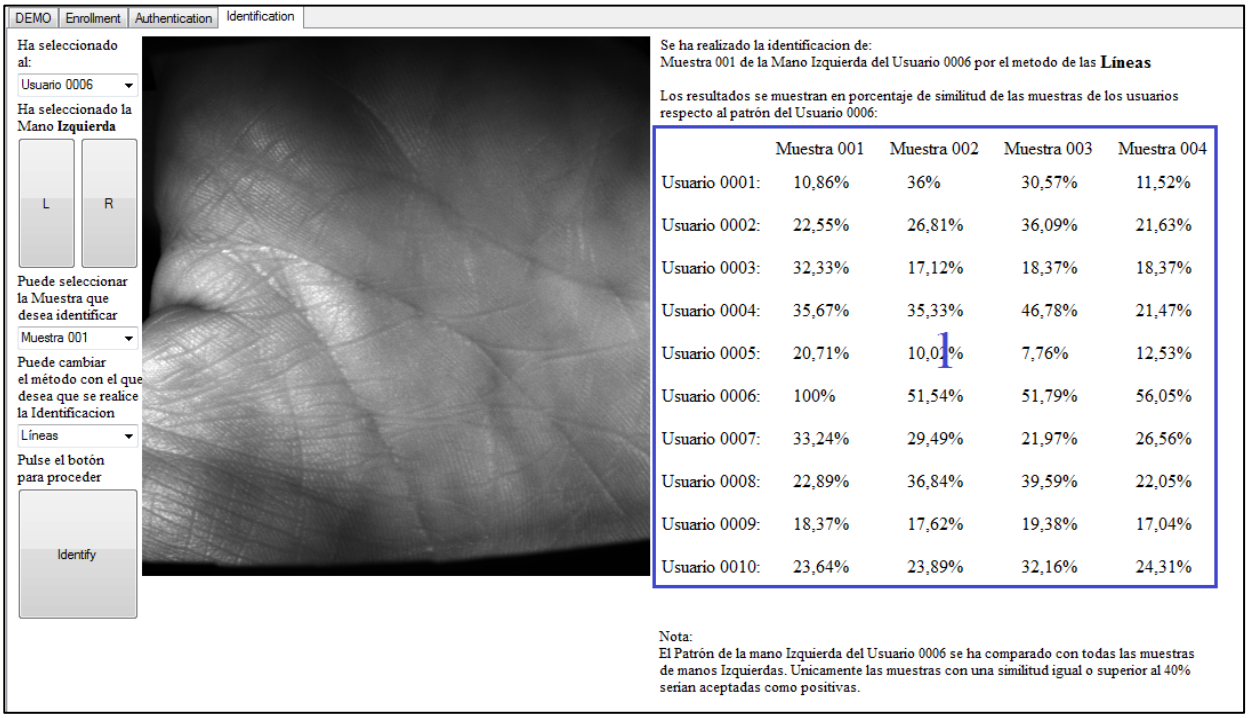

Figura 58. Identificación en la Aplicación Final

La funcionalidad de esta ventana se ajusta a la explicación de 5.2.4. Aunque el Mostrar Resultados ya aparece en menor medida en anteriores funcionalidades, en esta cobra especial importancia puesto a través de este proceso se muestran los resultados propios de esta funcionalidad, los porcentajes de semejanza de las muestras de la base de datos. Se comprueba que los resultados son idénticos a los de la aplicación sin estándar (Figura 27).

Con la comprobación de todos los procesos en todas las funcionalidades de la aplicación se confirma que la adaptación del estándar ha sido exitosa.

6.2.2 Interfaz de la Aplicación

Con las pruebas hechas en el apartado anterior queda más que comprobado el correcto funcionamiento de la interfaz del usuario. Por ello únicamente queda compararla con la de la aplicación original.

La aplicación original posee una interfaz pensada para el uso de un usuario más experto y con poco interés por la comodidad de este a la hora de trabajar con la aplicación. El trabajo con la aplicación se hace pesado, sobre todo por la existencia de numerosas ventanas para funciones que se han resumido en una sola ventana en la aplicación adaptada.

Hasta 5 ventanas son necesarias (Figura 59) para un proceso que en la aplicación adaptada se ha introducido en una ventana (Figura 60).

Por otro lado, la funcionalidad Verificación es más versátil en la Aplicación Adaptada que en la Original. En la versión C# es posible la verificación de una muestra contra cualquier otra de la base de datos (ver 5.2.3) mientras que en la Original solo es posible Verificar las muestras con otras del mismo usuario.

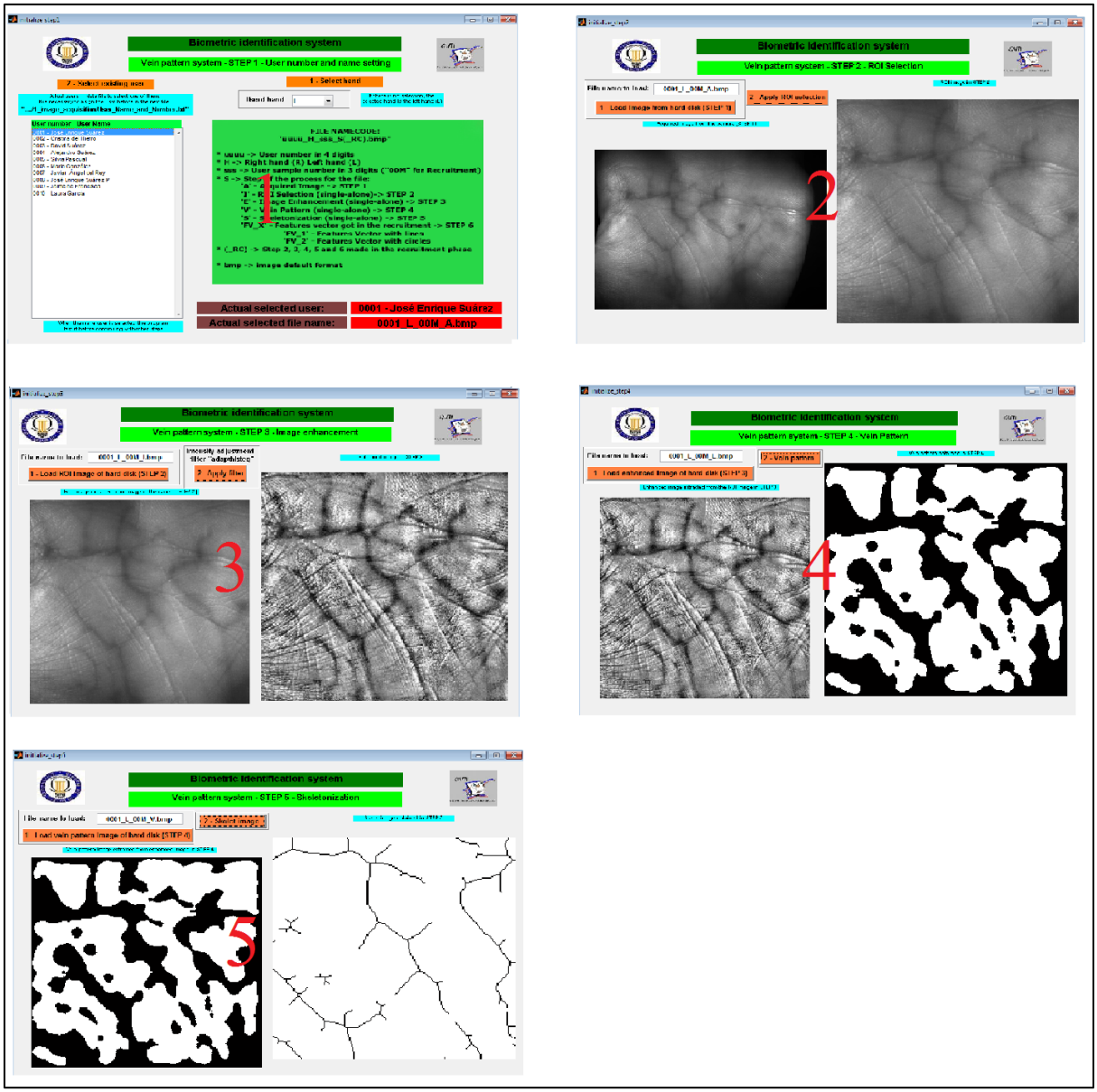

Figura 59. Ventanas de la Aplicación Original

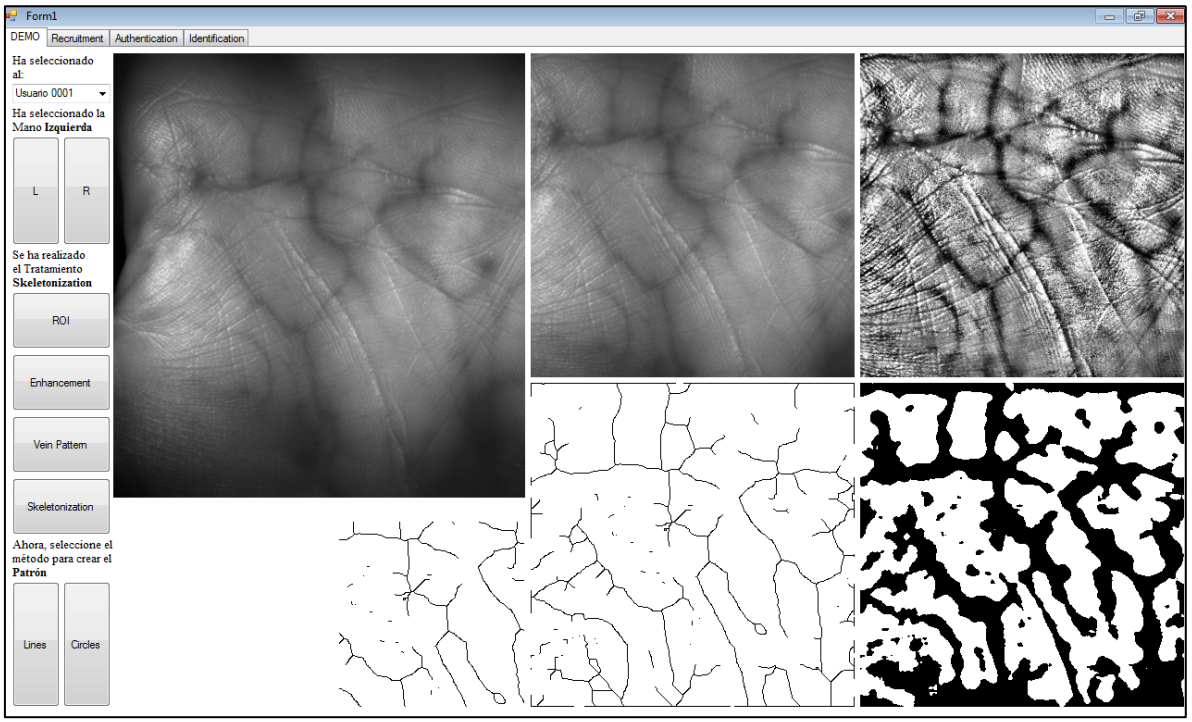

Figura 60. Simplificación en la interfaz en la Aplicación Adaptada

6.2.3 Rendimiento del Algoritmo

En este apartado se explica el método que se ha seguido para evaluar el rendimiento del algoritmo traducido.

La eficacia de cualquier método biométrico depende de su tasa de Falsa Aceptación (FMR) y de su tasa de Falso Rechazo (FNMR). El punto donde ambas tasas se cruzan se denomina Tasa de Error Igual (EER).

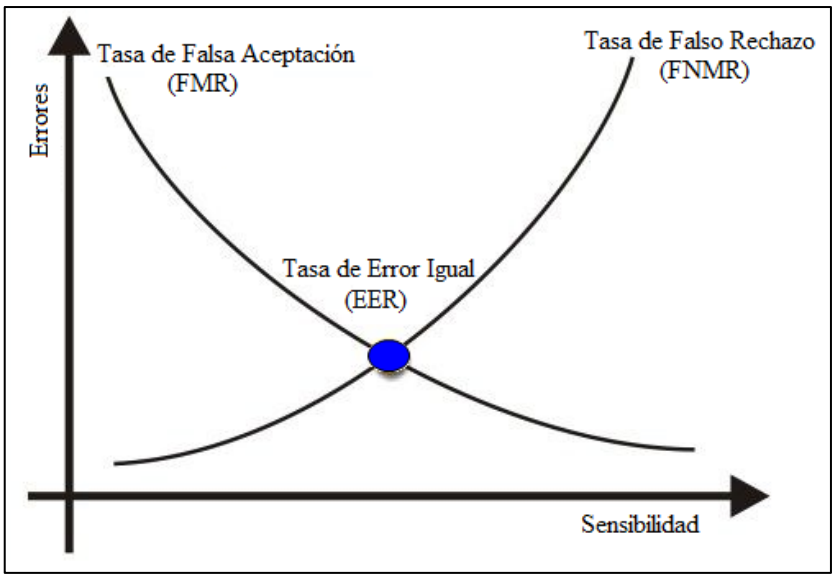

Figura 61. Cruce de tasas FMR y FNMR

Se suele considerar que cuanto más bajo se el EER, menos errores de identificación cometerá el sistema[27]. Pero, el rendimiento óptimo del sistema no siempre se dará para este punto. Dependiendo de en que condiciones de seguridad se utilice el sistema (más o menos restrictivas) puede llegar a ser interesante el desplazamiento a la izquierda o a la derecha del umbral de decisión. Como por ejemplo, tener más falsos rechazos pero con la seguridad de que todos los que consiguen superar la prueba del sistema son quienes dicen ser (el umbral de decisión se situaría a la derecha del umbral que proporciona el EER en la gráfica de la Figura 61).

Tras comprobar el funcionamiento del algoritmo en la aplicación ya adaptada al estándar se procedió a la evaluación del algoritmo. Se ha utilizado para ello la funcionalidad de Identificación de la aplicación, es la que única que proporciona datos cuantitativos en cuanto a la similitud de las muestras. Siguiendo el ejemplo de Dr. Suarez-Pascual, para la evaluación del algoritmo adaptado se han hecho las mismas pruebas que se hicieron para evaluar el algoritmo original en la aplicación MATLAB obteniendo además, de esta manera, la posibilidad de evaluar la calidad del porte del algoritmo.

A continuación se explica el proceso de selección de datos para la creación de las tasas.

En la base de datos de Manos se cuenta con 10 Usuarios con 4 Muestras para cada una de sus (2) Manos. Es decir:

10Usuarios X 4Muestras X 2Manos = 80Muestras

Para las pruebas de la aplicación se ha realizado la identificación de 1a muestra-patrón de cada usuario contra el resto de muestras de la misma mano del resto de usuarios. Al igual que se ha hecho para las pruebas de la aplicación MATLAB, la muestra patrón se ha seleccionado arbitrariamente entre las 4 disponibles, concretamente la muestra numero 1 de todos los usuarios se ha considerado como patrón. Por tanto, se enfrenta 1 Patrón contra 39 Muestras, y esto en cada usuario.

Ejemplo: Patrón de mano izquierda de usuario1, contra las otras 3 muestras de esta mano del usuario1 (no se incluye la muestra-patrón, esta identificación siempre da 100% de similitud) más 4 muestras de la mano izquierda de los otros 9 usuarios, (4 X 9 = 36)

Puesto que el procedimiento se realizó con 10 Usuarios para cada Mano y además con 2 Métodos (*Lines* y *Circles* 5.3.1.5) distintos, obteniéndose así:

39Comparaciones X 10Usuarios X 2Manos = 780Resulados

Se obtuvieron por tanto 780 Resultados para la evaluación de cada uno de los métodos.

Con los resultados obtenidos, para el cálculo de las tasas FNMR y FMR se ha tenido en cuenta las siguientes especificaciones:

- **FNMR**. Para hallar esta tasa únicamente son relevantes las muestras del usuario que se está identificando. Esto significa que se tienen 3 muestras (no se toma el resultado de la muestra-patrón contra muestra-patrón puesto que siempre es del 100% y falsearía los resultados) en cada una de las 10 Identificaciones (una por Usuario) y por 2 manos. Se obtienen así 60 muestras para el cálculo de FNMR y esto para cada uno de los dos Métodos (*Lines* y *Circles*).

- **FMR**. Para esta tasa son relevantes las muestras de los usuarios que NO sean del usuario que se está identificando. Por tanto se tienen, en cada identificación, 4 muestras por cada uno de los 9 usuarios y por cada mano. Esto resulta en 72 resultados en 10 Identificaciones, es decir, 720 resultados para el cálculo de FMR y esto por cada uno de los dos métodos (*Lines* y *Circles*).

Los datos obtenidos se distribuyen en tablas y se les aplican las especificaciones arriba explicadas.

METODO *Lines*

USUARIO 1 - Mano izquierda

Usuario/Muestra	1	2	3	4
1	100	41	63	63
2	12	25	30	19
3	19	20	17	15
4	31	33	29	34
5	2	6	12	12
6	11	34	35	31
7	17	18	18	18
8	21	24	22	21
9	25	27	27	28
10	35	48	49	49

USUARIO 1 - Mano derecha

Usuario/Muestra	1	2	3	4
1	100	34	27	79
2	37	20	37	32
3	14	14	22	6
4	23	20	38	21
5	6	17	7	5
6	17	21	23	17
7	29	15	28	15
8	8	4	4	6
9	7	15	16	22
10	5	6	0	15

METODO *Circles*

USUARIO 1 - Mano izquierda

Usuario/Muestra	1	2	3	4
1	100	70	28	30
2	0	21	0	4
3	28	34	34	34
4	14	7	11	0
5	27	11	27	0
6	4	8	0	0
7	65	47	53	53
8	63	69	69	63
9	11	11	30	28
10	59	0	23	10

USUARIO 1 - Mano derecha

Usuario/Muestra	1	2	3	4
1	100	18	24	96
2	0	31	27	23
3	25	10	18	54
4	16	13	33	27
5	13	8	12	26
6	19	24	5	18
7	36	37	28	34
8	32	0	0	0
9	6	13	18	18
10	33	37	1	31

Figura 62. Tablas con los datos de las Identificaciones

En la Figura 62 se representan las tablas que contienen los datos de las identificaciones a las que han sido sometidas las muestras del usuario 1. En estas se discriminan los datos con una sensibilidad del 30%. Esto quiere decir que se establece como umbral un 30% de similitud entre Patrón y Muestra y se observa como de eficaz es este umbral. El código de colores de las tablas señala:

- **Verde**. Muestras que deberían superar el umbral y lo superan.

- **Azul**. Muestras que deberían superar el umbral y NO lo superan (estas incrementan el FNMR).
- **Amarillo**. Muestras que NO deberían superar el umbral y NO lo superan.
- **Rojo**. Muestras que No deberían superar el umbral pero lo superan (estas incrementan el FMR)

Después del recuento de los resultados, la tasas obtenidas para la sensibilidad del 30% son:

- *Lines*. **FMR**=178 de 720 Muestras significativas y **FNMR**=7 de 60 Muestras significativas.
- *Circles*. **FMR**=215 de 720 Muestras significativas y **FNMR**=17 de 60 Muestras significativas.

El mismo proceso se ha seguido para todas las sensibilidades múltiplos del 10 (10%, 20%,…, 100%) y tras el recuento final de todas las muestras que contribuyen al **FMR** y **FNMR**, se han podido construir las siguientes gráficas.

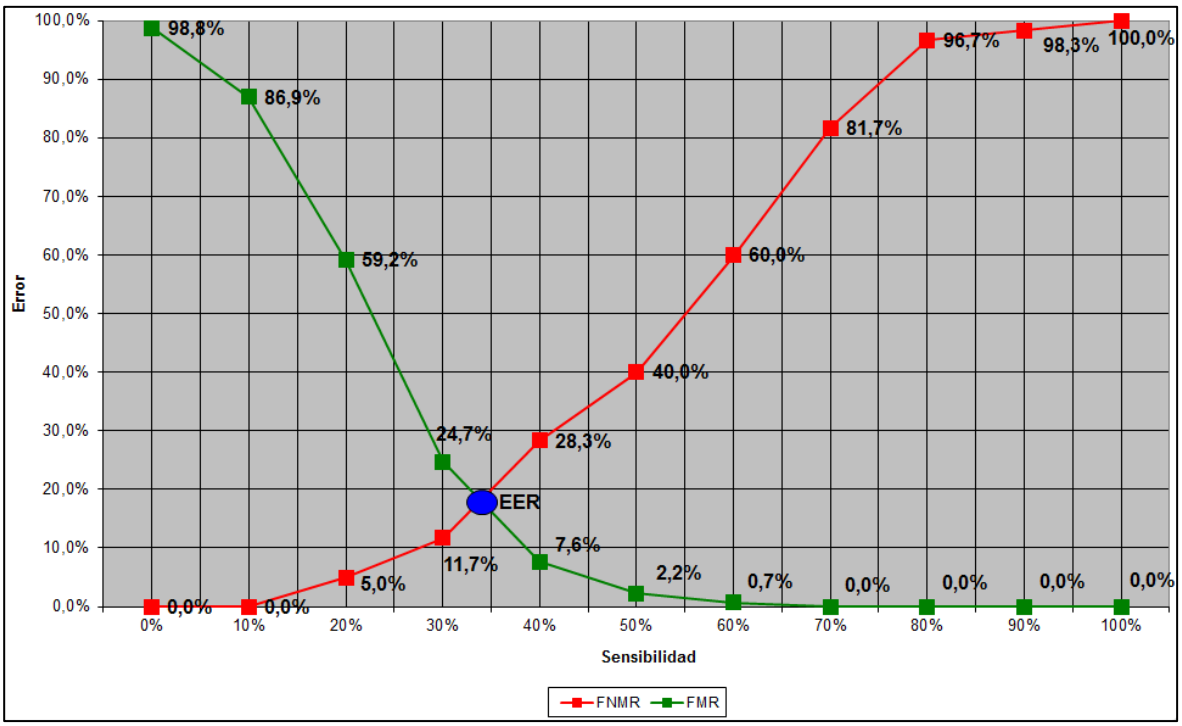

Figura 63. EER del Método *Lines* en la Aplicación C#

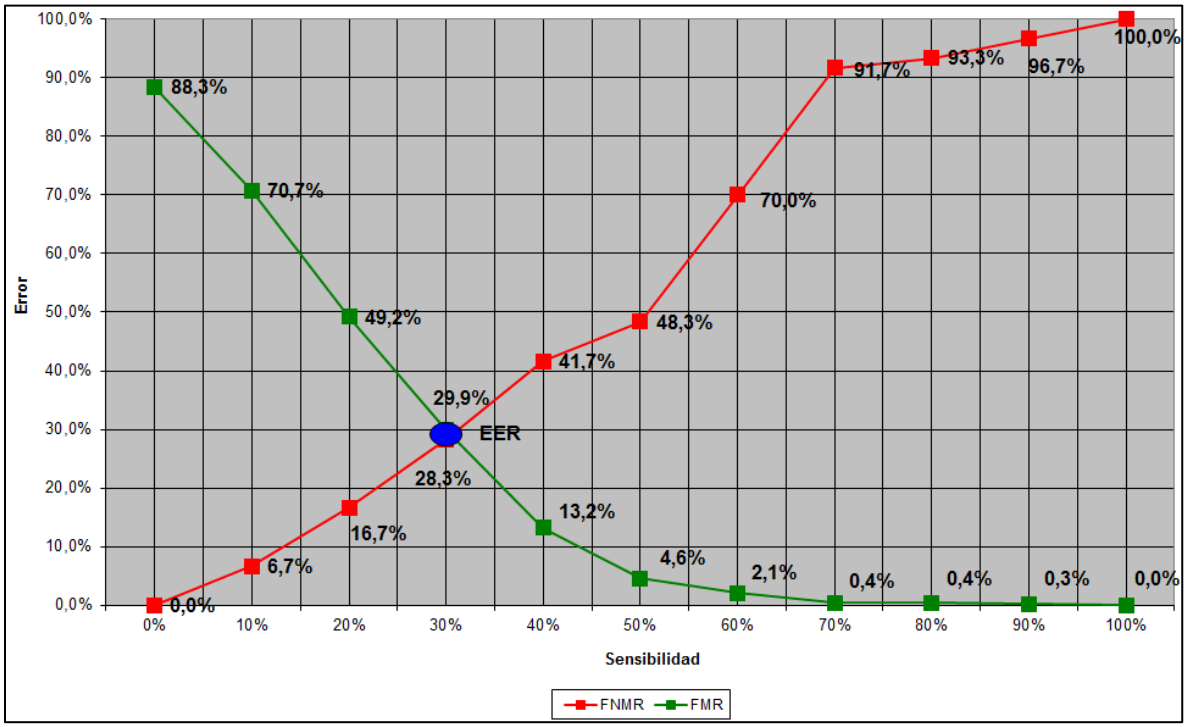

Figura 64. EER del Método *Circles* en la Aplicación C#

Analizando los resultados obtenidos se deduce que los umbrales de sensibilidad óptimos para los métodos *Lines* y *Circles* son 34% y 30% respectivamente. Con estos umbrales las tasas de fallo obtenidas son respectivamente para los métodos del 18% y 28%. Esto quiere decir que con los **parámetros actuales**, con este sistema se obtendrían 18 falsas aceptaciones y 18 falsos rechazos por cada 100 identificaciones según el método *Lines* y 30 falsas aceptaciones y rechazos según el método *Circles*.

Observando los porcentajes es evidente que el método *Lines* ofrece resultados mejores que el *Circles*, del orden del 10% (ver diferencias entre métodos en5.3.1.5). Aun así, una tasa de fallo del 18% no sería aceptable en un sistema biométrico comercial. En los sistemas de identificación vascular que actualmente se comercializan, las tasas de fallo son mucho más bajas (menores al 0,03% [28]), dejando así muy detrás al sistema aquí diseñado.

Por otro lado, en la comparación con los resultados de la aplicación original se obtiene que en la adaptación del algoritmo al nuevo lenguaje se ha perdido calidad (Figuras 65 y 66). Concretamente se trata de una pérdida de calidad del 7% en el EER del método *Lines* (del 11% de tasa de error en aplicación original al 18% de la aplicación C#) y 8% en el método *Circles* (del 20% en aplicación original al 28% en la aplicación C#).

Se estima que la pérdida de calidad radica en los distintos tipos de datos que manejan los lenguajes de origen y destino (apartado 5.3). Además es posible que gran parte de esta pérdida se produzca en los pasos modificados durante la traducción del algoritmo (apartado 5.3.1.3 y Figura 38 y apartado 5.3.1.5). Como se comenta en estos capítulos, de manera

inexplicable una adaptación rigurosa del algoritmo al nuevo lenguaje produce resultados lejos de los esperados por lo que se introdujeron cambios con tal de acercarse a la calidad de los resultados de la aplicación original.

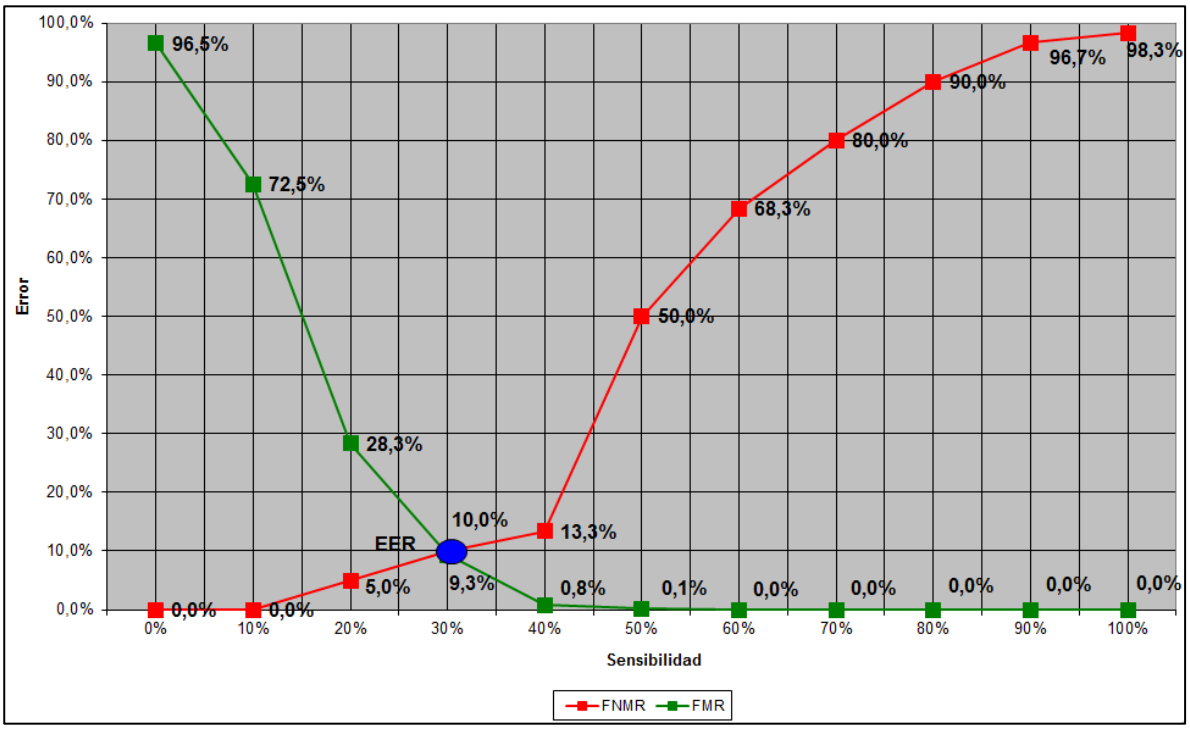

Figura 65. EER del Método *Lines* en la Aplicación Original

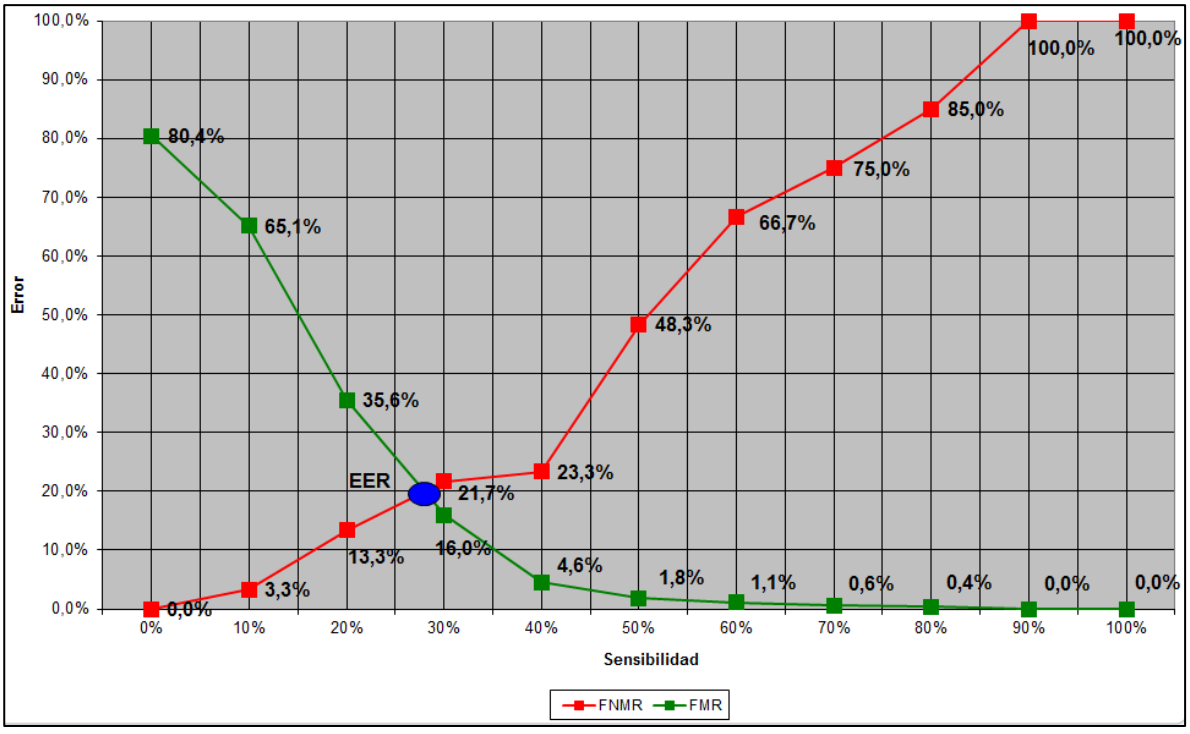

Figura 66. EER del Método *Circles* en la Aplicación Original

Técnicamente es posible mejorar los resultados del algoritmo, manteniendo la estructura del mismo. Los cambios que se pueden realizar son mínimos, y se trata de modificar los parámetros de los filtros que son aplicados a las imágenes durante el proceso Procesado de Imágenes. Los parámetros implementados durante a traducción eran los de la aplicación original con algunos, pequeños cambios justificados únicamente por comparación visual de resultados (apartado 6.1).

Usando la funcionalidad de Identificación y la discriminación de datos explicada en este capítulo como herramientas, se ha introducido cambios en los parámetros de los filtros que mejoran el rendimiento del algoritmo. Lamentablemente los cambios introducidos debían ser muy pequeños puesto que se ha comprobado que cambios sustanciosos hacen inservibles a las muestras. Ver resultados en las Figuras 67 y 68.

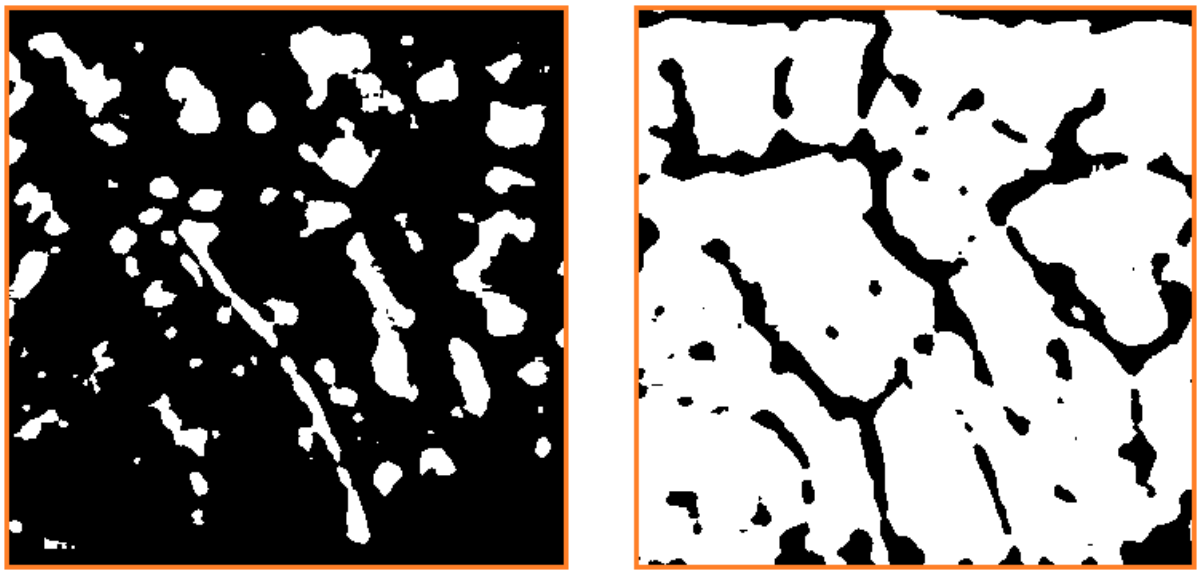

Figura 67. Destrucción de la muestras con cambios sustanciosos en los filtros (filtro de binarización de imagen 5.3.1.3)

Se representan en esta figura un aumento (izquierda) y una reducción del umbral de binarización de la función `ThresholdBinary`. Como se observa, un cambio de poco más del 10% (30 sobre 256) sobre la posición establecida como optima 130, provoca la destrucción de las muestras; se obtienen muestras casi idénticas en todos los usuarios con el umbral alto y muy pocos puntos de interés con el umbral bajo.

Figura 68. Destrucción de la muestras con cambios sustanciosos en los filtros (filtro mediana 5.3.1.3)

Se representa en la figura el resultado de aumentar considerablemente el parámetro del área de aplicación del filtro en la función `SmoothMedian`. Aunque a simple vista no parece que la muestra haya quedado inservible, se ha comprobado que con este valor del parámetro, el número de puntos de interés hallado se reduce casi a la mitad, respecto al obtenido con el ajuste que se considera óptimo de este paramento en la función, 9.

Los pequeños cambios finalmente introducidos implican cambios mínimos en los porcentajes de similitud de muestras por lo que las tasas de Falso Rechazo y Falsa Aceptación apenas varían algunas unidades. Finalmente observando los resultados con los cambios introducidos se obtuvieron las tasas de error arriba descritas, mejorando únicamente un 1% el resultado del método *Circles* hasta el 28% de fallo (desde el 29% tras la traducción original del algoritmo).

7 Conclusiones y Líneas de Futura investigación

7.1 Conclusiones

Al finalizar este trabajo se tiene una aplicación, completamente funcional y adaptada al estándar BioAPI, que simula el funcionamiento de un sistema de identificación biométrica por medio de venas de las manos.

Durante el desarrollo de la aplicación, las principales dificultades aparecidas siempre tenían como fondo el desconocimiento inicial, del alumno, de los lenguajes de programación usados en este trabajo. Especialmente complejo ha sido el proceso de traducción del algoritmo de tratamiento de imágenes, sin embargo una rigurosa planificación y la investigación de las posibles soluciones con la ayuda del tutor siempre ha dado sus frutos en la búsqueda de resultados.

La aplicación de prueba del algoritmo junto a la implementación del estándar desarrollados, solamente es un proyecto de investigación, sin previsiones comerciales. Puede servir de apoyo en el desarrollo de futuras aplicaciones con algoritmos nuevos, así como para cualquier proyecto que incluya el estándar BioAPI.

Como conclusión propia del alumno, es remarcable que el alumno partía desde un extremadamente limitado de lo que es la biometría y especialmente de las herramientas informáticas que se han utilizado en durante el desarrollo del trabajo. Sin embargo, termina el trabajo con conocimientos suficientes para poder seguir trabajando en otros proyectos relacionados con la biometría y/o las tecnologías aprendidas en el transcurso de este. Se considera por tanto que se han cumplido los objetivos tanto técnicos como personales propuestos al principio de esta memoria.

7.2 Líneas Futuras de Investigación

En este trabajo queda patente que es posible la adaptación al estándar de algoritmos creados con bibliotecas de código libre. Así mismo, es evidente que el algoritmo usado en este trabajo no es de primera calidad y debería ser mejorado en futuras versiones. Sobre todo es recomendable introducir mejoras en el paso de creación de patrones, lo que en este trabajo se ha denominado como técnicas *Lines* y *Circles*. Se sugiere la introducción de algoritmos más elaborados como el de Maio[29] o Jain[30], cuya efectividad ya ha sido comprobada en modalidades biométricas como identificación por huella dactilar, y que de igual manera pueden ser aplicados en esta modalidad. Estos algoritmos de búsqueda de minucias se espera que mejoren notablemente las tasas de error del sistema creado obtenidas en las pruebas. Por

otro lado, para obtener unos resultados más representativos sería necesario ampliar la base de datos de muestras con la que se hacen las pruebas.

Así mismo este trabajo es solo un paso más en la investigación y creación de algoritmos para esta modalidad biométrica. Junto al reconocimiento por árbol de venas de las manos, actualmente están en desarrollo sistemas que utilizan con el mismo fin las venas del dorso de las manos o de las muñecas. Como propuesta para el futuro, se abre la posibilidad de crear patrones 3D con las venas de toda la mano incrementando de esta manera la unicidad de las muestras. Para ello sería necesaria la creación de otro sistema con nuevos sensores de adquisición y también el desarrollo de nuevos algoritmos de tratamiento y comparación de muestras.

Durante el desarrollo de este trabajo, la norma del estándar BioAPI ha sufrido cambios muy importantes con la creación de nuevas funciones y eliminación de algunas que se han usado en la adaptación del código traducido en este TFG. Como una futura línea de investigación, sería interesante la adaptación del algoritmo a la nueva versión de la norma que resulte de estos cambios.

Bibliografía

[1] Nuevas técnicas de identificación http://seguridadbiometrica.over-blog.com/article-identificacion-vascular-62749701.html. Fecha de consulta: 10/08/2014

[2] Identificación vascular en Japón http://www.eltiempo.com/archivo/documento/MAM-1682364. Fecha de consulta: 10/08/2014

[3] Identificación Vascular en Reino Unido http://www.bbc.com/news/business-29062901. Fecha de consulta: 11/09/2014

[4] Directiva 95/46/CE, del Parlamento Europeo y del Consejo, de 24 de octubre. D.O.U.E. serie L, núm. 281, de 23 de Noviembre de 1995

[5] Ley Orgánica 15/1999, de 13 de Diciembre, de Protección de Datos de Carácter Personal. B.O.E. núm. 298, de 14 de diciembre de 1999

[6] Real Decreto 1720/2007, de 21 de diciembre, por el que se aprueba el Reglamento de desarrollo de la Ley Orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal. B.O.E. núm. 17, de 19 de enero de 2007

[7] Observatorio de la Seguridad de la Información de INTECO (sociedad estatal adscrita al Ministerio de Industria, Turismo y Comercio), "*Estudio sobre las tecnologías biométricas aplicadas a la seguridad"*, Diciembre 2011

[8] Biometría en el siglo XIV http://es.wikipedia.org/wiki/Biometría - Historia. Fecha de consulta: 18/08/2014

[9] Alphonse Bertillon, "*Identification Anthropométrique"*, París 1890. Consultado en: https://archive.org/details/identificationan00bert)

[10] Alphonse Bertillon http://es.wikipedia.org/wiki/Alphonse_Bertillon. Fecha de consulta: 18/08/2014

[11] Observatorio de la Seguridad de la Información de INTECO (sociedad estatal adscrita al Ministerio de Industria, Turismo y Comercio), *"Guía sobre tecnologías biométricas aplicadas a la seguridad"*, Octubre 2011

[12] Gemelos idénticos http://geneticsawareness.org/esgen/aprende-acerca-de-la-genetica/tiene-preguntas-sobre-la-genetica/los-gemelos-identicos-son-100-identicos-geneticamente. Fecha de consulta: 19/08/2014

[13] Face Recognition Grand Chalenge http://www.nist.gov/itl/iad/ig/frgc.cfm. Fecha de consulta: 19/08/2014

[14] Unicidad del árbol de venas de las manos http://tecnoweb.com/2014/06/fujitsu-lanza-un-sistema-para-pagar-con-las-venas-de-la-mano/. Fecha de consulta: 30/08/2014

[15] Esquema de un Sistema Biométrico http://revistasic.com/revista39/agorarevista_39.htm. Fecha de consulta: 30/08/2014

[16] ISO/IEC. "Information Technology -- Biometrics -- BioAPI for object oriented programming languages -- Part 3: C# implementation". ISO/IEC 30106-3. Fecha de publicación prevista: 22/12/2015

[17] Emgu CV http://www.emgu.com/wiki/index.php/Version_History - Emgu.CV-2.4.9_Beta. Fecha de consulta: 25/08/2014

[18] OpenCV http://en.wikipedia.org/wiki/OpenCV - History. Fecha de consulta: 25/08/2014

[19] Historia BioAPI http://redyseguridad.fi-p.unam.mx/proyectos/biometria/estandares/bioapi.html. Fecha de consulta: 26/08/2014

[20] Arquitectura BioAPI http://en.wikipedia.org/wiki/BioAPI. Fechas de consulta: 26/08/2014

[21] ISO/IEC. "Information technology -- Biometric data interchange formats -- Part 9: Vascular image data". ISO/IEC 19794-9:2007. Fecha de última revisión: 25/10/2011

[22] Instalando Emgu CV http://www.emgu.com/wiki/index.php/Download_And_Installation. Fecha de consulta: 02/09/2014

[23] Fuente del Software del TFG http://e5.onthehub.com/WebStore/ProductsByMajorVersionList.aspx?ws=8738733a-6d9b-e011-969d-0030487d8897&vsro=8. Fecha de consulta: 02/09/2014

[24] J. Enrique Suarez-Pascual, 7-Nov-2011, Tesis Dr. "Mecanismo de captura y procesado de imágenes de venas para identificación personal". Consejero: Dr. Raúl Sánchez Reíllo. Publicado por el departamento de Tecnología Electrónica de la Universidad Carlos III de Madrid

[25] Teoría sobre el Algoritmo Zhang-Suen http://rosettacode.org/wiki/Zhang-Suen_thinning_algorithm. Fecha de consulta: 13/09/2014

[26] Implementación del algoritmo Zhang-Suen en C++ http://opencv-code.com/quick-tips/implementation-of-thinning-algorithm-in-opencv/. Fecha de consulta: 13/09/2014

[27] Rendimiento de un Método Biométrico http://es.wikipedia.org/wiki/Biometr%C3%ADa#Funcionamiento_y_rendimiento. Fecha de consulta: 14/09/2014

[28] Hoja de características de lector de venas Hitachi H-1 http://www.kimaldi.com/productos/sistemas_biometricos/lectores_huella_digital_para_pc/lector_de_venas_del_dedo_hitachi_h_1. Fecha de consulta: 17/09/2014

[29] Algoritmo de Maio. Dario Maio, Davide Maltoni; "Direct Gray-Scale Minutiae Detection in Fingerprints", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 1, Enero 1997. Fecha de consulta: 18/09/2014

[30] Algoritmo de Jain. Anil Jain, Salil Prabhakar; “Fingerprint matching Using Minutiae and Texture Fatures”, International Conference on Image Processing pp. 282-285, Oct 7-10-2001. Fecha de consulta: 18/09/2014

Anexo A: Planificación y Presupuesto

En este anexo se realiza un desglose de actividades realizadas durante el transcurso del TFG y en base a este se realiza el cálculo de su coste.

A.1 Planificación

Se comentan aquí todas las fases explicadas en el capítulo 5:

Fase 1: Trabajo Previo

I. Instalación de Herramientas necesarias (MATLAB, Visual Studio y Bibliotecas Emgu CV) (7,5 horas)

II. Familiarización con las herramientas

   i. Estudio de tutoriales y video-tutoriales y realización de ejemplos con MATLAB, Visual Studio y Bibliotecas Emgu CV (17,5 horas)

   ii. Estudio del estándar y práctica con aplicaciones que ya lo tienen implementado (20 horas)

III. Tutorías (15 horas)

Fase 2: Diseño de la Aplicación

I. Diseño de la Aplicación Visual (20 horas)

Fase 3: Portado del Código

I. Procesado de Imágenes (85 horas)

II. Comparación de Imágenes (20 horas)

III. Procesos de Apoyo (25 horas)

Fase 4: Adaptación al Estándar

I. Adaptación de la Aplicación al Estándar (35 horas)

Fase 5: Pruebas en la Aplicación

I. Procesado de Imágenes (25 horas)

II. Comparación de Imágenes (15 horas)

Fase 6: Elaboración de la memoria

I. Redacción de la memoria (65 horas)

II. Corrección y maquetación (10 horas)

Las horas empleadas en corregir errores tras las Pruebas en la Aplicación están reflejadas en la fase de Portado del Código.

Tabla 1 - Desglose de tareas

FASES	HORAS EMPLEADAS
Trabajo Previo	60
Diseño de la Aplicación	20
Portado del Código	130
Adaptación al Estándar	35
Pruebas en la Aplicación	40
Elaboración de la Memoria	75
TOTAL	**360**

A.2 Presupuesto del Trabajo Fin de Grado

A.2.1 Costes materiales

Los materiales necesarios para la realización de este TFG han sido un ordenador, se recomienda de altas prestaciones para realizar los tratamiento de imágenes en un tiempo razonable, y las licencias de los programas utilizados. La licencia de Visual Studio al igual que la del sistema operativo utilizado (Windows 7 Ultimate) son gratuitas gracias a los acuerdos de la universidad explicados en 5.1.1. Las bibliotecas Emgu son de código libre por lo que son también gratuitas. En cuanto al software, el único reflejado en la tabla de costes debe ser la licencia de MATLAB puesto que fue cedida por el tutor para este trabajo. Considerando un periodo de amortización del ordenador de 3 años y teniendo en cuenta el tiempo del proyecto, los costes materiales quedan como se expone en la Tabla 2.

Tabla 2 – Costes Materiales

CONCEPTO	PRECIO (€)
Ordenador de altas prestaciones	100
Licencia MATLAB	30
TOTAL	**130**

A.2.2 Costes de personal

Este trabajo lo ha realizado un ingeniero con la supervisión de un jefe de proyecto.

Tabla 3 – Costes de Personal

OCUPACIÓN	HORAS	PRECIO/HORA	IMPORTE (€)
Jefe de proyecto	30	50	1.500
Ingeniero	360	30	10.800
TOTAL	**300**		**12.300**

A.2.3 Costes totales

Tabla 4 – Costes Totales

CONCEPTO	PRECIO (€)
Costes de materiales	130
Costes de personal	12.300
Costes indirectos (20%)	2.486
Subtotal	14.916
IVA (21%)	3.132.36
TOTAL	**18.048,36**

El coste total del proyecto es de DIECIOCHO MIL CUARENTA Y OCHO EUROS CON TREINTA Y SEIS CÉNTIMOS.

Leganés, 19 de Septiembre de 2014

Stanislav Hryhor